UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

TREBALL FI DE GRAU

Grau en Enginyeria Elèctrica

SEGUIDOR SOLAR AMB HARDWARE DE BAIX COST

Memòria i Annexos

Autor: Héctor Solà Vázquez
Director: Sergi Fillet Castella

Convocatòria: Maig 2018

Resum

El present treball de final de grau té per objecte la realització d'un primer prototip de seguidor solar amb hardware de baix cost, on es farà una sèrie de proves i assajos per tal de determinar la seva viabilitat. Aquest hardware estarà completament operatiu i ha de ser capaç de poder ser versàtil per així poder servir per altres experiments no realitzats en aquest document.

En aquest estudi s'escollirà el hardware i els components més adients que formaran part de cadascun dels diferents mètodes de seguiment solar i s'analitzaran la viabilitat d'implementació de cadascun. Els articles escollits, tot i ser de baix cost, seran seleccionats per tal de poder tenir la màxima qualitat i precisió.

Es realitzarà una estructura en la qual s'instal·laran les cel·les solars, els sensors i actuadors i que s'utilitzarà per anar provant les diverses programacions de seguiment solar. Aquesta estructura haurà de ser versàtil i fàcilment modificable i ampliable.

Es proposen diferents tipus de seguidors i s'analitzarà la viabilitat d'implementació dins dels components escollits. De cada mètode de seguiment proposat en aquest treball, s'analitzaran les programacions realitzades i la seva dificultat, les dades recollides en cadascun dels assajos i els avantatges i inconvenients. Finalitzant amb unes conclusions sobre l'estudi desenvolupat.

Resumen

El presente trabajo de final de grado tiene por objeto la realización de un primer prototipo de seguidor solar con hardware de bajo coste, donde se hará una serie de pruebas y ensayos para determinar su viabilidad. Este hardware estará completamente operativo y debe ser capaz de poder ser versátil para así poder servir para otros experimentos no realizados en este documento.

En este estudio se escogerá el hardware y los componentes adecuados que formarán parte de cada uno de los diferentes métodos de seguimiento solar y analizarán la viabilidad de implementación de cada uno. Los artículos escogidos, a pesar de ser de bajo coste, serán seleccionados para poder tener la máxima calidad y precisión.

Se realizará una estructura en la que se instalarán las celdas solares, los sensores y actuadores y que se utilizará para ir probando las diversas programaciones de seguimiento solar. Esta estructura deberá ser versátil y fácilmente modificable y ampliable.

Se proponen diferentes tipos de seguidores y se analizará la viabilidad de implementación dentro de los componentes elegidos. De cada método de seguimiento propuesto en este trabajo, se analizarán las programaciones realizadas y su dificultad, los datos recogidos en cada uno de los ensayos y las ventajas e inconvenientes. Finalizando con unas conclusiones sobre el estudio desarrollado.

Abstract

The purpose of this final degree project is to create a first prototype of a solar tracker with low cost hardware, where a series of tests and essay will be carried out to determine its viability. This hardware will be fully operational and must be able to be versatile so that it can be used for other experiments not realized in this document.

In this study, the appropriate hardware and components that will be part of each of the different solar tracking methods will be chosen and the viability of each one's implementation will be analysed. The chosen items, despite being low cost, will be selected to be of the highest quality and precision.

A structure will be made in which the solar cells, sensors and actuators will be installed and used to test the various solar tracking programs. This structure must be versatile and easily modifiable and expandable.

Different types of followers are proposed and the feasibility of implementation within the chosen components will be analysed. From each follow-up method proposed in this work, the programming made and its difficulty will be analysed, the data collected in each of the trials and the advantages and disadvantages. Finishing with some conclusions about the study developed.

Índex

1. Prefaci

Avui en dia, la automatització de processos electromecànics es realitza programant mitjançant diagrames de Gantt i després transferint la informació de la seqüencia al software del PCL pertinent.

En uns anys la programació tendirà a avançar cap a llenguatges destinats avui en dia al àmbit de la informàtica; com per exemple Java, C++, html, sql, php,...

1.1. Origen del treball i motivació

Plantejat el canvi que hi haurà en el món de la programació industrial, he cregut convenient aprendre a programar amb microcontroladors una aplicació dins del àmbit de l'electricitat.

La aplicació escollida es un seguidor solar, on es necessari treballar amb sensors, senyals analògiques i digitals, analitzar el funcionament de diferents motors i seleccionar els components més adequats per la aplicació. A més a més es un estudi que es pot anar complicant i plantejant millores com introduir perifèrics com pantalla LCD o exportant les dades a un PC.

Es un estudi que serveix per inicialitzar-se en la programació i que segons es va avançant es pot anar compactant, millorant i detallant el codi.

1.2. Abast

L'abast d'aquest treball de fi de grau es la creació d'un primer prototip en el qual assajar diferents programacions de seguidors solars, complint l'objectiu d'utilitzar articles de baix cost en la seva construcció.

Aquest ha de complir els requisits següents:

- Prioritzar els components de baix cost
- Que pugi ser utilitzat en diversos mètodes de seguiment tant d'un eix com de dos.
- Completament operatiu.

Per assegurar aquests requisits es faran un seguit de proves per tal de corroborar la viabilitat del hardware dissenyat.

1.3. Requeriments previs

Els requeriments necessaris per comprendre l'estudi son els següents:

- Coneixements basics d'electricitat
- Coneixements basics de programació
- Funcionament dels diversos actuadors de petita potencia que es solen utilitzar en projectes d'aquestes dimensions (motors pas a pas, servomotors,...)
- Coneixements dels tipus de senyals i el seu tractament

2. Introducció

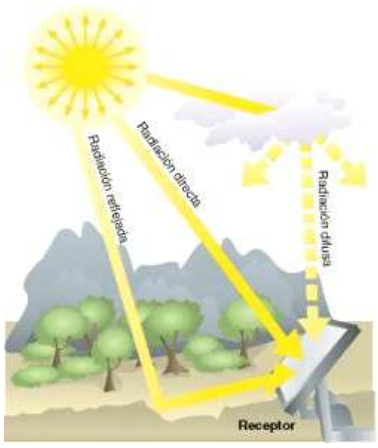
Un seguidor solar es un mecanisme que de forma automàtica orienta a la superfície receptora de manera que els rajos del Sol incideixin de forma perpendicular sobre aquesta. Existeixen una gran varietat de prototips al mercat que permeten el seu ús per a diverses funcions.

Aquests seguidors poden ser controlats des de manualment fins a un control mitjançant una programació o amb un programa que determina la posició del Sol.

El ús més comú que se li dona als seguidors solars es la generació d'energia elèctrica per després autoconsumirla o vendre-la; tot i així existeixen altres usos menys comuns com per exemple alimentar senyals de tràfic lumíniques , per a robots autòmats, generació d'energia tèrmica o per fer ombra.

Es important la aprofitar energia emesa per el Sol ja que es una font il·limitada i renovable. Aquesta energia es pot utilitzar perquè la radiació solar emesa per el Sol, es transmet en formes d'ones electromagnètiques. La llum solar esta composta de fotons a diferents longituds d'ona, i per tant, amb diferents energies.

La energia provinent del Sol pot incidir a la superfície receptora de manera directa, reflectida o difusa. La radiació solar directe es quan els raigs del sol incideixen directament, aquest fenomen passa quan el cel no esta cobert. La radiació reflectida ocorre quan la superfície receptora rep els raigs de manera indirecta gracies a que aquests raigs han rebotat en un altre element, com pot ser l'aigua, neu o col·lectors destinats dirigir l'energia solar a un punt (energia tèrmica de alta temperatura). Veure Imatge 3.1.



Imatge 3.1. Tipus de radiacions solars (Font calculationsolar.com)

En el cas de l'energia tèrmica, per les plaques solars es fa passar un circuit tancat de líquid en el qual es transfereix l'energia calorífica. Aquest circuit passa per un dipòsit d'aigua on es cedeix part de la calor. Aquests dipòsits poden ser de consum directe, o simplement de recolzament d'una caldera.

Existeixen instal·lacions petites on l'únic objectiu es el subministre d'aigua calenta sanitària (ACS), on nomes hi ha un sol dipòsit amb una sortida a consum i amb un estudi tècnic es pot determinar el volum d'acumulació del dipòsit i la superfície de plaques a instal·lar per poder donar cobertura a unes certes necessitats. El percentatge de potencia calorífica coberta per les plaques solars es diu fracció solar (%).

En les instal·lacions de major potencia, on la superfície de plaques solars tèrmiques es inviable, s'utilitzen calderes per cobrir part de les necessitats tèrmiques demanades, on es solen tenir dos dipòsits diferenciats; un per l'energia calorífica generada per les plaques solars tèrmiques i l'altre per la provinent de les calderes. En aquest últim dipòsit es on estarien les diferents sortides d'ACS i calefacció.

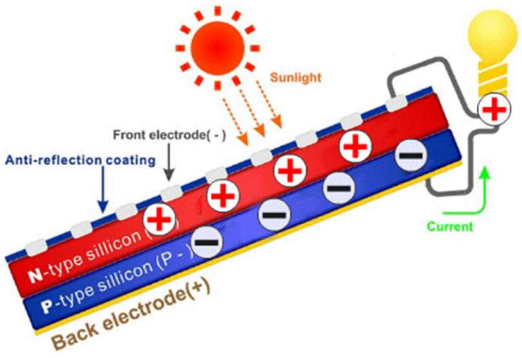
Cal dir que també aquesta energia tèrmica es pot transformar en energia elèctrica.

Depenent del ús que es faci de l'energia elèctrica generada, ja sigui convertint energia calorífica en elèctrica o obtenint-la directament, existeixen dos tipus d'instal·lacions en funció de si es una instal·lació aïllades o connectada a la xarxa.

Les instal·lacions aïllades consten d'un regulador, bateries i inversors de corrent. La funció del regulador es interrompre la connexió dels panells solars amb les bateries per així evitar que es produeixi una sobrecarrega i poder augmentar la vida útil d'aquestes. El inversor serà utilitzat per passar de la tensió donada per el regulador a 230 V i 50 Hz, per així poder alimentar els receptors. Les bateries s'encarreguen de emmagatzemar l'energia elèctrica no utilitzada per a futures necessitats.

En contrapartida, les instal·lacions connectades a la xarxa solament tenen incorporat un inversor i un comptador, on la energia generada es subministrada a la xarxa i on l'operador de mercat la gestiona.

La generació d'energia elèctrica mitjançant plaques solars degut al efecte fotovoltaic. Com que les cèl·lules solars tenen dos semiconductors amb diferents propietats, al absorbir part de la llum solar s'alliberen electrons (electrons lliures), i gracies al moviment d'aquests entre els semiconductors apareix una diferencia de potencial.



Imatge 3.2. Efecte fotovoltaic (Font: www.certificadosenergeticos.com)

La llum rebuda a la cèl·lula solar pot arribar directament del sol, de manera difusa traves dels núvols o reflectida; on tots tres casos la radiació solar aporta energia però l'aportació major de potencia serà donada per la radiació solar directa. Això significa que la placa solar haurà de estar orientada en tot moment perpendicularment a la llum provinent del sol, que implica que l'angle entre la superfície de la placa i el vector de la radiació solar sigui de 90º. Per tant, per aprofitar al màxim l'energia solar s'implementen mètodes de seguiment, en busca de obtenció de la potencia màxima en cada instant de temps.

Actualment, els tipus de places més comuns son els panells monocristal·lins i policristal·lins, aquests últims son els que més s'utilitzen ja que el cost de fabricació es menor i els rendiments ronden els 75-80 %.

Els mètodes de seguiment solar es poden classificar depenent al nombre d'eixos que es fa el control o el mètode utilitzat per poder fer el seguiment.

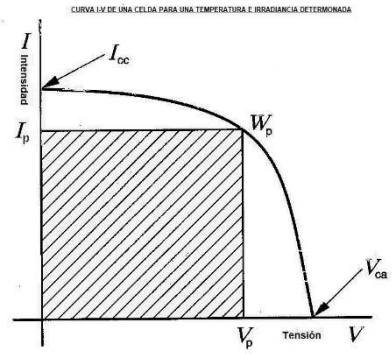
El control per eixos pot ser variant els angles azimutal i/o zenital, donant a tres possibilitats diferents. L'angle azimutal es l'angle que es forma entre el Sol i el sud en la horitzontal del observador, es mesura en sentit horari. L'angle zenital es l'angle format entre la línia d'unió del Sol amb l'observador, respecte la línia horitzontal del terra.

L'altre opció es utilitzant dos eixos perpendiculars entre ells.

Els mètodes emprats per fer el control dels eixos solen ser:

- Seguidor de punt de màxima potencia (MPPT) :

  Les corbes de funcionament de les plaques solars no son lineals i per tant per aconseguir treballar amb el punt de màxima potencia existeix una determinada tensió; i aquesta varia depenent de la temperatura i la irradiància solar. L'aspecte de la corba de Intensitat-Tensió es la següent:



Imatge 3.3. Curva I-V de una cel·la fotovoltaica (Font: www.areatecnologia.com)

Aquest seguidor s'implementa en el regulador, on es aquest qui determina mitjançant algoritmes els punts òptims de tensió i intensitat per tal d'obtenir la màxima potencia de sortida. Un d'aquests mètodes es basa en la modificació de la tensió de sortida del regulador i llegir els valors de potencia da la entrada del inversor, depenent si ha hagut un augment o una disminució de la potencia d'aquest, la variació de la tensió realitzada es correcte.

En el següent enllaç, inclòs en la bibliografia, s'expliquen els diferents mètodes de seguiment MPPT.

http://bibing.us.es/proyectos/abreproy/70172/fichero/Resumen.pdf

El seguidor del punt de màxima potencia es el que obté la major potencia en cada instant en funció de la velocitat de variació de posició, te uns rendiments superiors al 95%.

- Seguidor emprant sensors de lluminositat LDR:

Els seguidors que utilitzen sensors fotoresistors son més simples que els anteriorment nombrats, consten de 3 o 4 sensors de llum situats al voltant del panell solar i que sensat la resistència en cada moment es determina cap on hi ha més radiació solar.

Aquest tipus de seguidor es el més emprat amb hardware de baix cost, ja que la programació es senzilla. Aquest mètode no s'utilitza per la generació d'energia elèctrica en les centrals fotovoltaiques, sinó que son més comunes les aplicacions en projectes de petita potencia; com robots autònoms. El rendiment d'aquest mètode de seguiment es inferior al MPPT ja que no treballa amb valors de potencia i els sensors tenen desviacions, fet que s'hagin de calibrar. El rendiment ronda el 75%

- Seguidor amb base de dades de la posició solar en funció del temps

Aquest mètode de seguiment te com a base un llistat de dades que indican la posició solar per a una certa ubicació i en funció del temps. Aquestes dades es poden consultar diferents pagines webs i poden ser de diferent indole. Es poden utilitzar dades històriques, de previsió de la posició en el futur o dades instantànies.

El major inconvenient d'aquest mètode es la impossibilitat d'evitar les pertorbacions i ombres que puguin aparèixer i fer disminuir notablement el rendiment de les plaques solars.

Algunes pagines per poder consultar aquestes dades son:

- https://www.sunearthtools.com/dp/tools/pos_sun.php?lang=es
- http://www.tierradelazaro.com/calculo-de-varios-datos-del-tiempo-y-de-la-posicion-del-sol
- http://www.solartopo.com/home-es.htm

- Seguidor del punt de màxima tensió:

En algunes aplicacions es interesant la recerca del punt de màxima tensió, normalment es en aplicacions de petita potencia. Aquests seguidors prenen una lectura de la tensió subministrada i introduint petites pertorbacions analitzen la variació de la tensió i depenent d'aquesta inverteixen el moviment fer per les plaques solars.

3. Anàlisis dels mètodes de seguiment solar

Es proposen diferents mètodes de seguidors solars que s'analitzaran els avantatges i inconvenients de cadascun. Els mètodes son els següents:

- Cerca del punt de màxima tensió
- Seguidor mitjançant la posició solar
- Seguidor mitjançant fotoresistències

Tots aquests mètodes tindran un control a dos eixos per així poder tenir uns millors resultats.

3.1. Mètode 1. Cerca del punt de màxima tensió

Aquest mètode de seguiment consisteix en avaluar la tensió generada per el conjunt de plaques solars en un punt i comparar-lo posteriorment amb altres punts propers, fent així que la placa sempre estigui enfocant al punt de màxima irradiació solar on la tensió es major. Aquest mètode no segueix el Sol amb precisió, ja que podrien aparèixer possibles ombres, boira o núvols i el seguidor evitaria aquests punts.

La cerca del punt de màxima tensió sembla útil sobretot als matins i en els dies on el Sol està tapat, això es degut a que ens aquests moments la radiació solar difusa sumada a amb la reflectida pot ser major que la radiació directa. A més a més, un seguidor a dos eixos pot enfocar el vector normal de la superfície de les plaques cap a l'alba o a la posta de Sol i així aprofitar la radiació solar en aquests moments més crítics per una placa fixe. Com la potencia generada per les plaques solars es veu afectada per la temperatura, quan sent aquesta menor la potencia generada augmenta, es pot aconseguir un gran rendiment.

Un dels inconvenients d'aquest seguidor es la possibilitat de poder quedar orientant cap a un reflex i quedar "atrapat" en aquesta posició.

Aquest mètode no es el més recomanat per la generació de energia elèctrica per la distribució o autoconsum; sinó per a petites instal·lacions de corrent continu o per projectes autònoms de petita potencia com robots.

3.2. Mètode 2. Seguidor mitjançant la posició solar

El segon mètode es basa en tenir un llistat de la situació del Sol en el cel en funció del temps, i que el nostre el conjunt de les plaques el segueixi durant el llarg del dia. Depenent del lloc d'instal·lació de les

plaques solars, existeixen bases de dades amb els històrics de les posicions solars i previsions de futur. Només es tractaria de treballar amb aquestes dades.

Aquest es l'únic mètode que no te en compte les ombres i altres pertorbacions que poden aparèixer. Obtenim precisió de seguiment solar sacrificant l'energia obtinguda (ja sigui tèrmica o elèctrica).

El seguidor solar mitjançant una base de dades es útil per fer estudis i investigacions sobre la radiació solar i per poder fer ombra en llocs que interessi; per exemple un para-sol autòmat que amb la radiació solar rebuda pugui girar per tal de fer ombra on estiguin les tovalloles.

3.3. Mètode 3. Seguidor mitjançant fotoresistències

El mètode consisteix en tenir quatre fotoresistències LDR situades en el centre dels costats de l'estructura i fer que el sistema equilibri els nivells de llum incident a cadascun d'ells. Aquest sistema es semblant al primer però amb diferents senyals d'entrada.

A priori, els resultats esperats d'aquest mètode ha de ser semblant al primer però lleugerament inferior, ja que no treballem amb la tensió que generen les plaques.

Les fotoresistències tenen valors màxim i mínim de resistència en funció de la llum rebuda, dins d'un mateix model de sensor LDR aquestes poden variar; per tant, es necessita calibrar-los inicialment per tant que tots tinguin la mateixa caiguda de tensió en les mateixes condicions. Un error en aquest aspecte pot generar que l'algoritme de seguiment faci que la placa solar no es posicioni correctament i que al no tenir cap altra forma de corroborar la correcta situació d'aquestes, el rendiment sigui inferior al esperat.

Es un mètode simple que per a petits projectes domèstics es molt utilitzat.

4. Selecció del hardware de baix cost

Es proposen diferents tipus de hardware de baix cost que s'analitzaran per tal d'escollir quin d'ells compleix millor amb els requisits de programació per cada mètode de seguiment mencionat en el apartat 3 d'aquesta memòria.

4.1. Plaques de desenvolupament Arduino

Arduino es una plaques de hardware lliure, on els esquemes i diagrames son d'accés públic i per tant es poden crear per un mateix en comptes de comprar-les. Existeix una gran quantitat de programacions, biblioteques, manuals d'ús i vídeos didàctics dels diferents compon (sensors, actuadors,..) per tal de facilitar el treballar amb d'aquests.

Aquestes plaques Arduino consten de pins digitals i analògics els quals poden configurar-se tant com a entrades o sortides i operar a 3,3 V i/o 5 V en funció del model escollit.

En funció d'unes senyals d'entrada dels sensors, actua donant unes senyals de sortida als actuadors en funció de la programació que s'hagin pujat al hardware.

La programació en el software Arduino es fàcil i intuïtiva, consta de tres parts identificades i flexibles a l'hora de treballar. Aquest software treballa configurant els pins i després fer el programa creat en bucle.

Els preus d'aquestes plaques oscil·len a la pagina web oficial entre 16-65€ en funció del model escollit (sense taxes), i en pagines webs com Aliexpress o Amazon es poder trobar per preus a partir de 6€.

Per tant, es una bona opció per realitzar els programes dels diferents mètodes de seguidors solars, amb una programació senzilla i flexible on l'entorn de desenvolupament afavoreix l'objectiu del treball.

Imatge 4.1. Placa Arduino UNO rev3 (Font: www.arduino.cc)

4.2. Microcontroladors PIC's

Un microcontrolador PIC es un circuit integrat que te una memòria EEPROM. Aquests microcontroladors tenen unes dimensions petites i existeixen una gran varietat d'encapsulats.

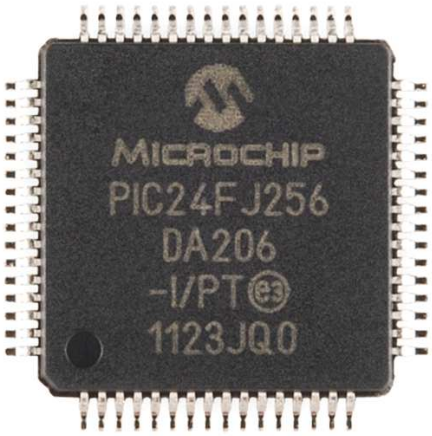
Consten de 6 a 84 pins depenent del model escollit, que poden treballar com entrades o sortides analògiques o digitals. Els microcontroladors PIC es divideixen en tres grans grups depenent de la amplada de bits del bus.

Existeixen una gran varietat de programadors i de llenguatges de programació per aquests tipus de microcontroladors. El codi d'aquests controladors es molt compacte ja que utilitza jocs d'instruccions reduïts (RISC). La programació es més complicada que Arduino, ja que es necessita una base de coneixements de programació.

Els preus d'aquests circuits integrals ronden entre 1-25€.

Les avantatges del circuit integral PIC es que al ser d'unes dimensions reduïdes i existir una gran varietat, es pot acoblar a qualsevol circuit ja muntat. I que als ser els programes compactes, aquests son més ràpids a l'hora d'executar-se.

Donada la complicació al programar i al fet de que els avantatges que ens aporta no son necessaris, descartem la opció de realitzar les programacions del nostre seguidor solar.



Imatge 4.2. Circuit integrat PIC24FJ26 (Font: www.sparkfun.com)

4.3. Placa computadora Raspberry Pi

Raspberry Pi es una placa que conté tots els components que sol tenir un ordinador però de poca potencia. Te connectors USB, HDMI, Wifi, Bluetooth, Ethernet, a més a més de pins per poder connectar perifèrics, entre altres.

Per poder treballar amb Raspberry Pi s'ha d'instal·lar el sistema operatiu. Raspbian es la opció recomanada per el fabricant, que es una plataforma idònia per programar amb Python, el qual ja ve integrat en el sistema operatiu.

La programació es realitza amb llenguatge Python (existeixen altres opcions). Aquest es flexible i ràpid, ja que en poques línies de codi es pot programar compacte.

El preu d'aquesta placa esta sobre els 50€ a Mediamarkt.

Raspberry Pi es una bona opció amb la qual programar els seguidors solars proposats, es una placa amb versatilitat on i que te opcions de complementar-ho amb les funcions d'un ordinador comú. El preu es major que els anteriors hardware.

Imatge 4.3. Placa Raspberry Pi 3 Model B (Font: www.raspberrypi.org

4.4. Controladors lògics programables (PLC)

Un PLC es un controlador que consta de entrades, sortides i un display.

Es poden programar amb portes lògiques o diagrames de contactes, aquest últim llenguatge es la millor opció de programació per algú amb més coneixements d'electricitat que d'electrònica o informàtica.

El major inconvenient son els preus d'aquests components, ja que a més a més s'ha de comprar mòduls acoblables a per tal de tenir més entrades, sortides i altres funcions especials.

El PLC més comú es el LOGO! de Siemens, el qual te un preu aproximat de 150€; fet que ens fa descartar aquesta opció.

Imatge 4.4. Mòdul Logic Programable Siemens Logo! 8 (Font: https://www.siemens.com)

4.5. Comparació i selecció del Hardware

Els requisits a complir per el hardware escollit son:

- Minimitzar el cost
- Facilitat de programació
- Quantitat d'entrades analògiques i digitals (PWM)
- Facilitat de trobar informació relativa al hardware escollit
- Software de programació propi

En la següent taula es valoren els requisits dels diferents hardware amb un numero entre el 1 i el 3, complint perfectament i no complint els requisits; respectivament.

	ARDUINO	PIC	RASPBERRY PI	PLC
PREU	1	1	2	3
FACILITAT DE PROGRAMACIÓ	2	3	2	1
QUANTITAT DE PINS DE CONNEXIÓ DISPONIBLES	1	2	2	3
FORUMS DE SUPORT	1	3	1	2
SOFTWARE DEDICAT A LA PROGRAMACIÓ	1	3	2	1
TOTAL	6	12	9	10

Taula 4.1. Comparació de requisits entre els diferents hardware proposats (Font: Varies)

El sumatori de les valoracions individuals determina el software a escollir, en aquest cas serà Arduino ja que es el que menys puntuació a tingut (per tant, es el que més compleix els requisits necessaris)

Entre Arduino i Raspberry Pi, Arduino es una placa pensada principalment per fer projectes d'electrònica on la programació es fàcil i intuïtiva amb un preu de compra baix. En canvi, Raspberry Pi, no esta pensat especialment per programar i tenint un preu major.

Es descarten els PIC i els PLC's, els primers per la complicació de programació i els segons per l'alt preu que tenen. A més a més els Inputs/Outputs disponibles son molt reduïts, i per fer una ampliació d'aquest s'ha d'invertir en més components.

Per tant, la elecció de hardware es Arduino.

4.6. Elecció del model de Arduino

Existeixen una gran varietat de models Arduino, depenent les necessitats del treball a realitzar s'escollirà el més adient.

Les necessitats que demanen els programes dels seguidors solar són pins analògics per a poder censar les diferents senyals provinents dels sensors (tensió, corrent,...). També són necessaris pins digitals per a controlar els motors que composaran el seguidor solar a dos eixos.

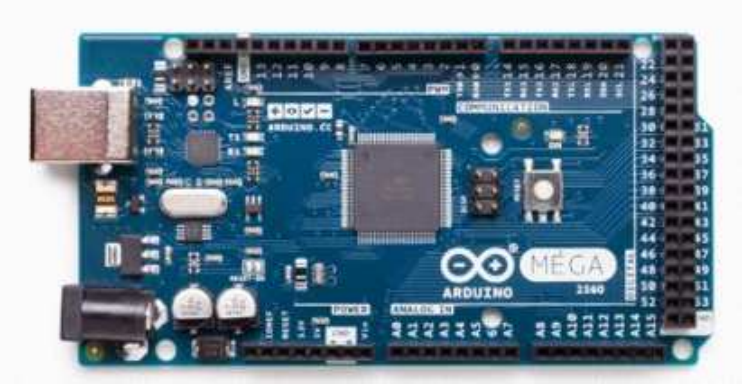
Amb aquestes característiques les plaques que s'adapten millor són:

- Arduino Uno
- Arduino Mega
- Arduino Leonardo

	MEGA 2560 REV3	LEONARDO	UNO REV3
MICROCONTROLLER	ATmega2560	ATmega32u4	ATmega328P
OPERATING VOLTAGE	5 V	5 V	5 V
INPUT VOLTAGE (RECOMMENDED)	7-12 V	7-12 V	7-12 V
DIGITAL I/O PINS	54	20	14
PWM CHANNELS	15	7	6
ANALOG INPUT PINS	16	12	6
FLASH MEMORY	256 KB of which 8 KB used by bootloader	32 KB of which 4 KB used by bootloader	32 KB of which 0.5 KB used by bootloader
CLOCK SPEED	16 MHz	16 MHz	16 MHz

Taula 4.1. Extracte de especificacions tècniques Arduino (Font: www.arduino.cc

S'escull la placa Arduino Mega 2560 Rev3 per ser la més potent respecte les altres proposades. El fet de disposar de més pins suposa la possibilitat de comparar alhora diferents mètodes de seguiment i/o utilitzar varis motors o pantalles LCD. Els 256 kB de memòria Flash permet tenir un gran llistat de dades en forma de matriu per al programa del mètode 2. Per tant, Arduino Mega 2560 Rev3 es el hardware escollit per realitzar l'estudi dels diferents mètodes de seguiment.



Imatge 4.5. Placa Arduino Mega 2560 rev3 (Font: www.arduino.cc)

4.7. Entorn de programació Arduino

El apartat 4.7 ha set extret i traduït del apartat 1.7.1 del següent document:

https://uvadoc.uva.es/bitstream/10324/17049/1/TFG-P-361.pdf

Arduino utilitza un llenguatge de programació anomenat Processing / Wiring i la seva sintaxi és semblant a la de C i similars. Permet crear un carregador d'arrencada o "bootloader" per iniciar les funcions del microcontrolador. L'entorn de desenvolupament utilitzat en aquests casos és "Arduino IDE". Basat en Java, ofereix un editor de text i una consola de depuració, al costat de les opcions de carregar el microcontrolador desenvolupat per mitjà d'un cable USB-micro USB.

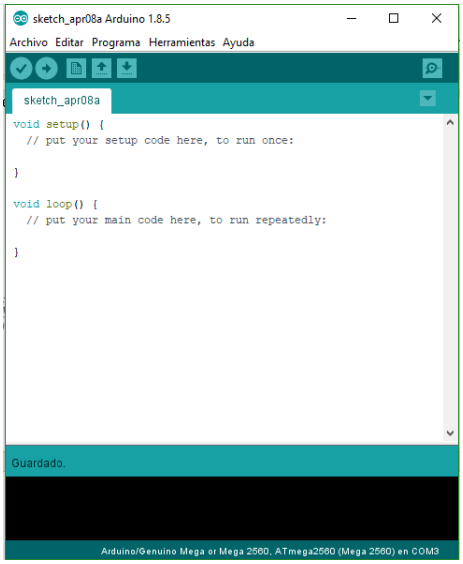
L'entorn de desenvolupament en Arduino (IDE) és l'encarregat de la gestió de la connexió entre el PC i el maquinari d'Arduino per tal d'establir una comunicació entre ells per mitjà de la càrrega de programes. L'IDE d'Arduino es compon de:

- Un editor de text, on escriure el codi del programa.
- Una àrea de missatges, a través del qual l'usuari tindrà constància en tot moment dels processos que es trobin en execució, errors de codi, problemes de comunicació, etc.
- Una consola de text, mitjançant la qual es permet la comunicació amb el maquinari d'Arduino.
- Una barra d'eines, on serà possible accedir a una sèrie de menús i als botons amb accés directe a les principals funcionalitats d'Arduino.

A través de la IDE d'Arduino, es pot escriure el codi del programa programari i crear el que es coneix per "sketch" (programa). El "sketch" permet la comunicació amb la placa Arduino. Aquests programes són escrits en l'editor de text, el qual admet les possibilitats de tallar, enganxar, buscar i reemplaçar text.

Un sketch té dues funcions clau:

- void setup (): Funció que només s'executa una vegada, serveix per inicialitzar les variables i la configuració de la placa.
- void loop (): És el cos principal del sketch. S'executa contínuament fins que s'apagui la placa.



Imatge 4.6. Arduino IDE (Font pròpia)

A l'àrea de missatges es mostra, tant la informació mentre es carreguen els programes, com els possibles problemes que es tinguin a l'hora de compilar.

Abans de començar, hem de triar a l'IDE quin serà el Arduino que farem servir i el port USB on està connectat.

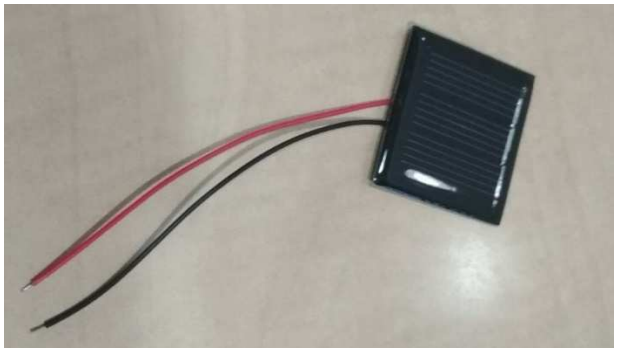
En el annex B2 s'adjunta un manual de programació amb les comandes bàsiques i la sintaxis de llenguatge de programació de Arduino.

5. Selecció de components

5.1. Plaques solars

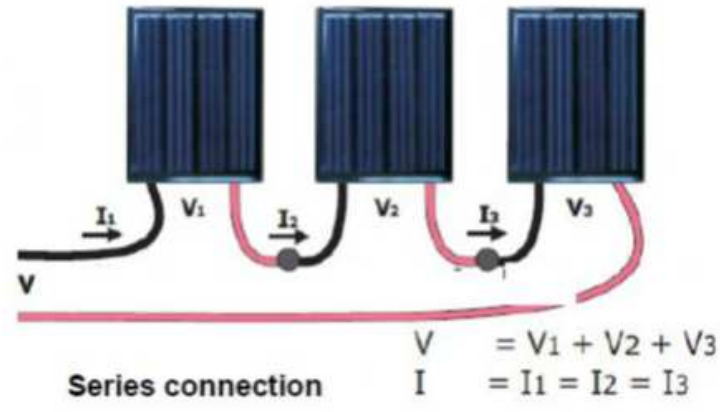
La tensió màxima dels pins de Arduino Mega escollit es de 5 V. Valor a tenir en compte a l'hora d'escollir les plaques. En cas de superar aquesta tensió màxima s'hauria de fer un divisor de tensió i en la programació tenir-ho en compte.

En el nostre cas s'ha escollit un paquet de quatre cèl·lules fotovoltaiques model C-0138 de la marca *Cebekit*. Les característiques tècniques d'aquestes plaques son unes dimensions de 40 x 40 mm, amb una tensió màxima per placa de 1,5 V DC, 100 mA i 150 mW en buit i amb 1,0 V DC i 75 mA en carrega.



Imatge 5.1. Celula solar (Font propia)

S'utilitzarà una placa com a referencia de tensió sent orientada cap al Sud i amb una inclinació determinada el terra. La configuració de les altres 3 cèl·lules serà en sèrie, per així sumar les tensions de cada una.



Imatge 5.2 Connexionat de les cel·les solars (Font: C-0138 Datasheet)

Les característiques de tècniques de les plaques s'adjunten en el Annex A2.

5.2. Sensors

La tensió generada per les plaques es censarà connectant el negatiu a terra i el positiu a una entrada analògica, així es tindrà la lectura de la tensió en cada moment. Per tant, els únics sensors necessaris per els tres mètodes de seguiment son els LDR.

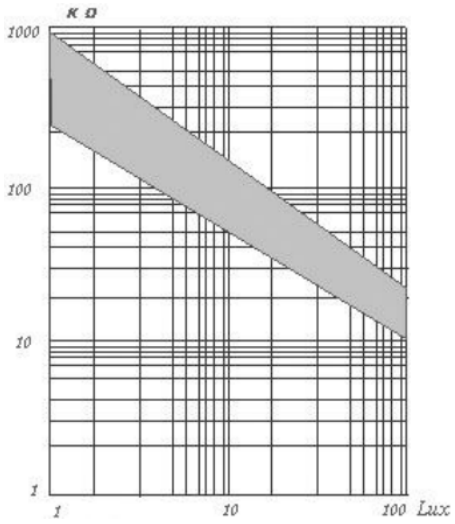
Una fotoresistència es una resistència feta d'un material semiconductor en el que la conductància varia amb la variació de la llum. Depenent del model presenta una resistència diferent en il·luminació màxima i en la foscor. Les prestacions i avantatges que ens proporcionen es les diferents característiques de espectre (Lux vs Resistencia), la petita mida , la alta sensibilitat i la resposta rapida.

Per a que Arduino pugui calcular el valor d'aquesta es necessari realitzar un divisor de tensió. Per tant, es posarà una resistència en sèrie amb el sensor LDR de valor adequat i la senyal d'entrada a Arduino serà el punt entre ells.

Com l'objectiu del mètode de seguiment solar mitjançant LDR es comparar la tensió entre els dos fotoresistors, una opció podria ser la de alimentar un LDR a +5 V DC i posar en sèrie l'altre LDR i connectat a terra. En el punt entre ambos sensors seria la senyal d'entrada a Arduino; aquesta, amb una alimentació de 5 V i en condicions de perfecta orientació ha de ser de 2,5 V DC. En aquest mètode el avantatge es la reducció de d'entrades a Arduino passant a utilitzar la meitat. En ambos casos s'ha de fer el calibratge de les fotoresistències, s'ha de comparar els valors de tensió donats per una mateixa llum incident i compensar la diferencia entre elles a la programació.

Al tractar-se d'un seguidor de baix cost ens decantarem per la opció més assequible. El model escollit per tant es el GL5539, aquest sensor LDR te aquestes característiques de resistència en funció de la llum incident:

- Resistència a la llum a 10 Lux (a 25 ºC) 50 ~ 100 kΩ
- Resistència a la foscor a 0 Lux + 5 MΩ

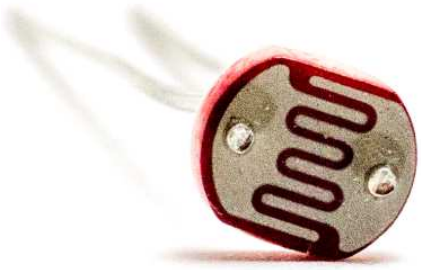


Imatge 5.3. Relació llum indicent i resistència de la fotoresistencia (Font: Datasheet GL55)

El preu unitari d'aquest sensor fotoresistor oscil·la entre els 0,015 € i 0,025 € en pagines com Aliexpress, Ebay o Amazon. Altres sensors de lluminositat en distribuïdors habituals superen sense problemes els 0,80 € per unitat. Per exemple:

http://es.farnell.com/w/c/optoelectronica-visualizacion/fotorresistencias-ldr

Al escollir el mètode de divisor de tensió utilitzant els dos sensors del mateix eix, no cal determinar la resistència que s'hauria de posar per tal d'aconseguir unes lectures precises.



Imatge 5.4. Fotoresistencia LDR model GL5539 (Font: inven.es)

5.3. Actuadors

En aquests diferents mètodes de seguidors solars, els únics actuadors necessaris seran els dos motors encarregats de fer el seguiment per a cada eix.

La principal característica del motor a escollir ha de ser la alta precisió en la posició, ja que es un projecte on els moviments a realitzar son molt curts, màxima força a velocitats baixes i que tinguin la possibilitat de quedar-se fixes en una posició mantenint-la sense mecanismes complicats.

Els diferents motors son:

- Motor DC
- Motor Brushless
- Servomotor
- Motor pas a pas

Els que tenen més precisió i més força a baixes velocitats son els motors pas a pas, degut a que depenent com s'alimentin les bobines s'aconsegueixen moviments molt petits. Alguns motors porten inclòs una reductora.

Les altres opcions també podrien ser viables si s'incorporés una reductora, però aquest fet incrementaria el preu del actuador; fet per el qual descartem els altres motors.

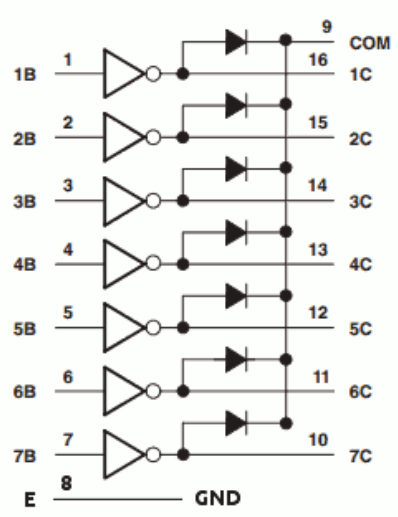
El motor pas a pas escollit es el 28BYJ-48 amb el que treballarem a mitjos passos, on un pas son 45 º. Com aquest motor te acoblat un engranatge reductor amb una relació de 1:64; això significa una precisió de moviment de 0,703 º.

Aquest motor necessita incorporar una placa de control, la més recomanada es la ULN2003A.

Imatge 5.5. Motor 28BYJ-48 + ULN2003 (Font: www.nooraziz.com)

Consisteix de un circuit integrat format per 7 transistors Darlington, 4 leds i els pins d'entrada i sortida. Per a cada fase del motor 28BYJ-48 s'utilitza un transistor Darlington. Aquest driver de sortida te un guany elevat degut a que es connecten dos transistors bipolars en cascada multiplicant el guany de

cadascun d'ells. Gracies a això es poden controlar cargues d'una potencia alta amb corrents d'entrada petites. Els leds indican quines fases estan sent alimentades.



Imatge 5.6. Esquema del circuit ULN2003 (Font: ULN2003A Datasheet)

Els motors pas a pas estan formats per dues parts, el estator i el rotor. El estator es la part fixe del motor, te unes osques i es on estan situades les bobines.

El rotor poc girar lliurement en el estator. Quan s'imanten les bobines una a continuació de l'altre, produeixen un desplaçament angular.

En funció de la alimentació de les bobines es pot aconseguir aprofitar millor les característiques dels motors pas a pas en funció de l'aplicació que se li donarà. Existeixen 3 tipus de moviments:

Alimentació de bobina en bobina.

	BOBINA 1	BOBINA 2	BOBINA 3	BOBINA 4
1	HIGH	LOW	LOW	LOW
2	LOW	HIGH	LOW	LOW
3	LOW	LOW	HIGH	LOW
4	LOW	LOW	LOW	HIGH

Taula 5.7. Moviment de pas complert

	BOBINA 1	BOBINA 2	BOBINA 3	BOBINA 4
1	HIGH	HIGH	LOW	LOW
2	LOW	HIGH	HIGH	LOW
3	LOW	LOW	HIGH	HIGH
4	HIGH	LOW	LOW	HIGH

Taula 5.8. Moviment de parell motor màxim

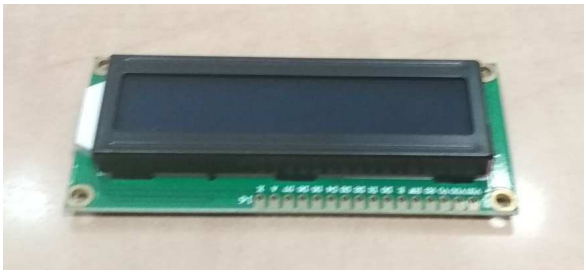
	BOBINA 1	BOBINA 2	BOBINA 3	BOBINA 4
1	HIGH	LOW	LOW	LOW
2	HIGH	HIGH	LOW	LOW
3	LOW	HIGH	LOW	LOW
4	LOW	HIGH	HIGH	LOW
5	LOW	LOW	HIGH	LOW
6	LOW	LOW	HIGH	HIGH
7	LOW	LOW	LOW	HIGH
8	HIGH	LOW	LOW	HIGH

Taula 5.9. Moviment de mitjos passos

En funció quin d'aquests tres mètodes d'alimentació de les bobines dels motors pas a pas s'aconsegueix diferents característiques. En aquest treball s'utilitzarà el ordre descrit a la taula 5.7.

5.4. Display

Es disposarà d'un display de 16 caràcters i 2 línies a la qual poder enviar informació del projecte. S'haurà de comprovar la disponibilitat de pins per poder connectar-la a la placa Arduino. El display seleccionat es el IIC 1602 LCD de 16 x 2.



Imatge 5.10. Display IIC 1602 LCD de 16 x 2 (Font pròpia)

5.5. Eines

Les eines necessàries per la realització d'aquest projecte son:

- Soldador elèctric de estany
- Alicates de punta fina
- Pelacables
- Trepant elèctric
- Pistola de silicona calenta
- Altres

6. Disseny i dimensionament de l'estructura

L'estructura es la part mecànica on s'ubicaran les cel·les solars i on els actuadors s'encarregaran de fer moure-les per tal de tenir la màxima incidència solar. Depenent del disseny de l'estructura, tota la programació i les diferents proves que es poden realitzar quedarien limitats o no es podran efectuar; per tant, el disseny ha de ser versàtil i senzill en el que poder carregar diferents codis.

La elecció més important del sistema es determinar el sistema de eixos a utilitzar. Existeixen seguidors d'eix polar, d'eix horitzontal, azimutal, zenital, de dos eixos,....

El disseny de l'estructura escollit es un sistema de 2 eixos perpendiculars entre ells; un s'encarregarà del moviment Nord-Sud i l'altre del Est-Oest. Aquest disseny ens permet començar primer construir-lo amb 1 eix per fer les proves pertinents i posteriorment ampliar-lo amb el segon eix.

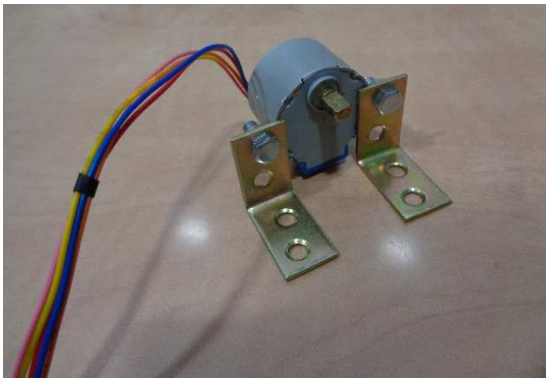
Els actuadors seleccionats son de petita potencia i per tant, per a cada un dels eixos de gir, s'instal·laran 2 units per el eix del motor. Per reduir les forces i inèrcies que puguin apareixen a l'estructura, el disseny minimitzarà el pes seleccionant materials els quals la seva densitat sigui baixa, bona resistència al aigua i que pugui suportar el pes de les cel·les solars.

Es descarten materials metàl·lics i derivats de la fusta ja que augmentarien el pes de la estructura considerablement i aportarien massa inèrcia. Per tant, el material ha de ser plàstic. El plàstic escollit es el metacrilat.

Els materials utilitzats per construir el seguidor son els següents:

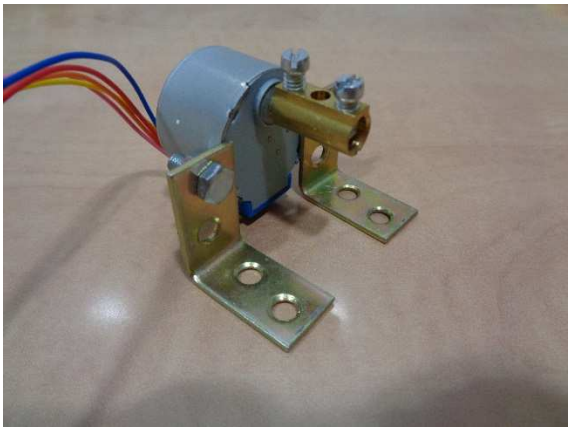
- Motors pas a pas 28BYJ-48 i plaques controladores ULN2003A
- Placa de metacrilat DinA4 de 3 mm transparent.
- Regleta de connexió de 16 mm^2
- Escaires bricomatades de 20 mm
- Cables de connexió
- Petit material com fil d'estany, barra de silicona, cargols, ...

Les escaires s'utilitzaran com a suport dels motors, una per a cada orella de cada motor; quedant de la següent forma:



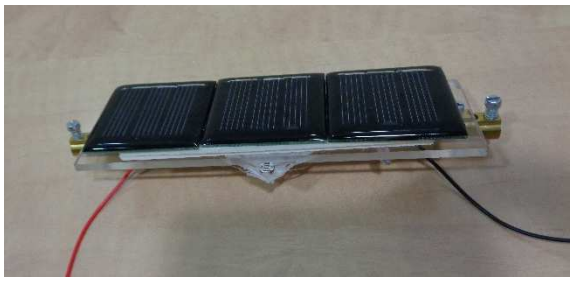
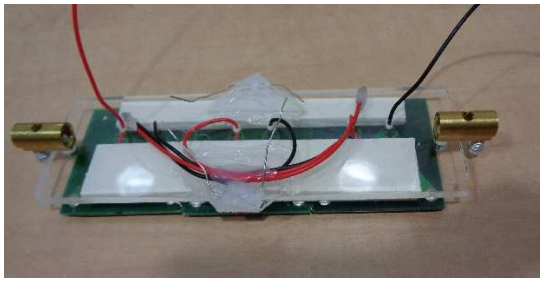
Imatge 6.1. Motor pas a pas amb les escaires (Font pròpia)

Ja amb l'ancoratge dels motors realitzat, es començarà tallant una secció rectangular de la placa de metacrilat amb unes mesures de 180 x 50 mm, en aquesta lamina es on es posaran les cel·les solars. Després s'eliminarà el plàstic de les regletes de connexió de 16 mm^2 per tal d'amarrar l'eix de cada motor amb un dels dos cargols. L'altre cargol anirà unit a la placa, i per això s'ha de realitzar un forat amb un trepant elèctric o un soldador just al mig dels laterals de 50 mm de la placa de metacrilat tallada prèviament.



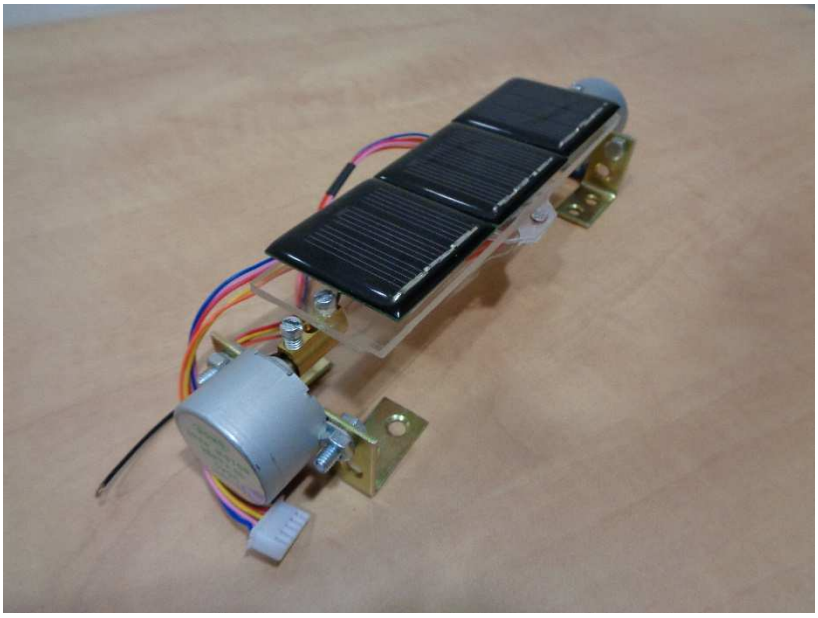
Imatge 6.2. Motor pas a pas amb la regleta al eix (Font pròpia)

Un cop units els motors pas a pas amb la placa de metacrilat, cargolarem les escaires al motor per la part exterior d'aquests i posteriorment hem de assegurar-nos de que girin correctament i que no es produeixin problemes de alineació entre els dos motors de cada eix. Per realitzar aquesta prova s'utilitzarà la programació corresponent en l'apartat 7.1 d'aquest document on s'explica els passos a seguir. En cas de donar problemes de alineació s'han d'ajustar tots els cargols fins aconseguir un resultat on el fregament produït sigui el menor possible.

Imatge 6.3. Placa de metacrilat amb les cel·les solars (Font pròpia)

Ja amb els motors units i en funcionament es realitzaran els forats a la placa de metacrilat per tal de passar tot el cablejat de les plaques i sensors LDR. Aquests sensors aniran centrats en la distribució de les plaques.

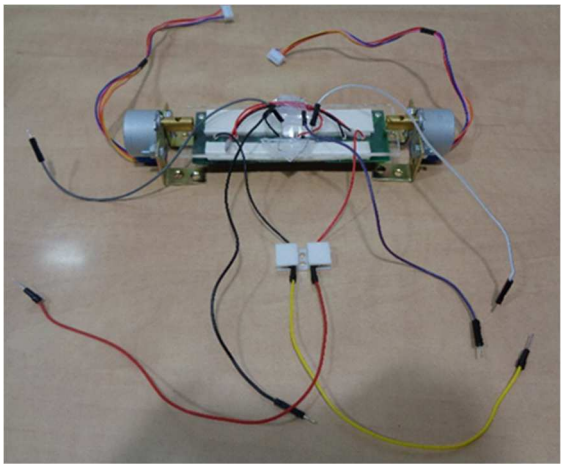


Imatge 6.4. Hardware de 1 eix(Font pròpia)

El següent pas es fer el connexionat sèrie de les plaques, utilitzant el soldador elèctric i el fil d'estany el soldaran el positiu d'una placa amb el negatiu de la següent fins només deixar dos cables solts. Aquests seran els que donaran la senyal al controlador Arduino.

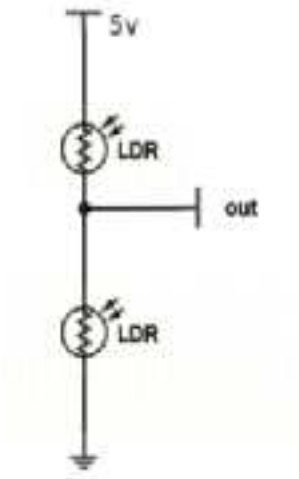
Al escollir les dues fotoresistències que s'utilitzaran, primer de tot es realitzarà la prova del apartat 7.1 per tal de determinar el valor de calibratge necessari a tenir en compte per les posteriors proves.

Per realitzar la instal·lació de les fotoresistències LDR, es fan els forats i es dona un punt de silicona calenta amb la pistola per la part inferior, per així assegurar que les fotoresistències no es mouen i que no ocorren falsos contactes entre les potes i que puguin alterar la senyal.



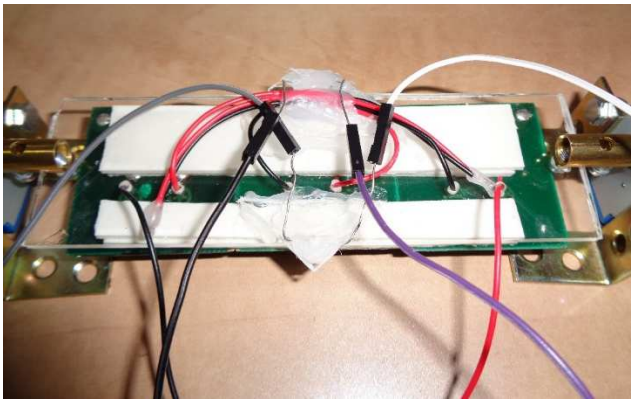
Imatge 6.5. Connexionat dels LDR i cel·les (Font pròpia)

L'esquema de connexió de les fotoresistències es el següent:



Imatge 6.6. Esquema de connexió de les fotoresistencies (Font pròpia).

Amb aquest esquema de connexionat de les fotoresistències LDR, el sistema estarà equilibrat quan la senyal d'entrada a Arduino sigui de 512 bytes o 2,5 V, o sigui, que la resistència de ambos LDR sigui equivalent produint la mateixa caiguda de tensió a cadascuna.

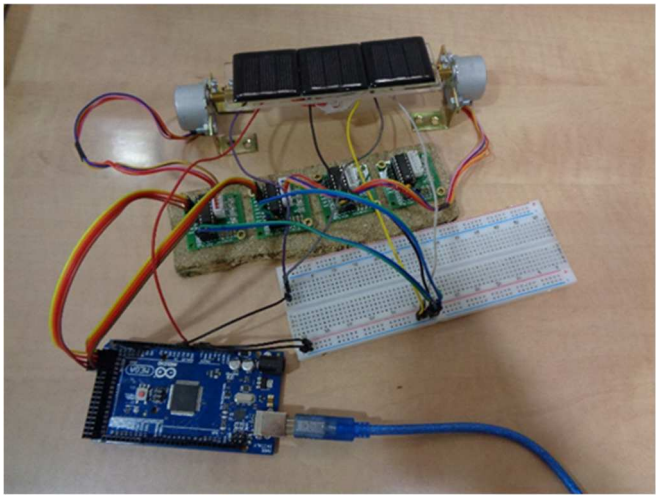


Imatge 6.7. Detall connexió LDR (Font pròpia)

Els cables dels LDR de color blanc i negre corresponen a +5 V i GNG, respectivament; el gris i el lila s'uneixen a la protoboard on d'aquí també s'envia la senyal a Arduino.

Dels pins +5 V i GND alimentem la protoboard i des d'aquí a les plaques de control ULN2003A, on el negatiu es el pin de la esquerra (-) i el positiu el pin de la dreta (+). Les sortides IN1-4 de cada placa de control dels motors es connectaran a els pins d'Arduino 53, 51, 49 i 47; 52, 50, 48 i 46 respectivament.

Ja tindríem el hardware seguidor de 1 eix preparat per provar els diferents mètodes de seguiment. En el hardware de 2 eixos aquesta part de la estructura seria la superior.



Imatge 6.8. Hardware per provar programacions d'un eix (Font pròpia)

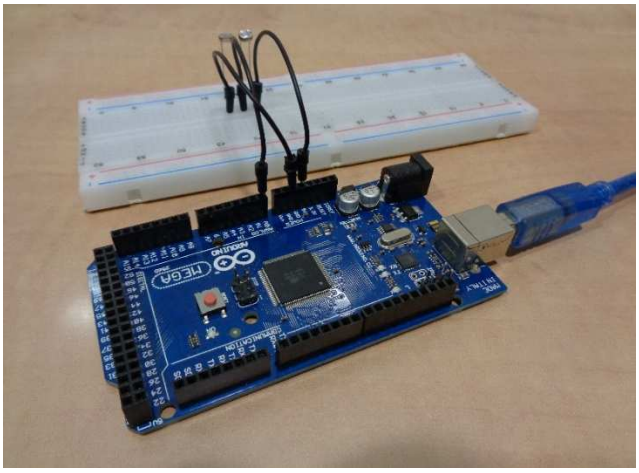
Aquesta instal·lació serà la mateixa per totes les proves que es faran; tant per llum artificial o natural, o utilitzant com a sensor les fotoresistències o les cel·les solars.

7. Programació Arduino

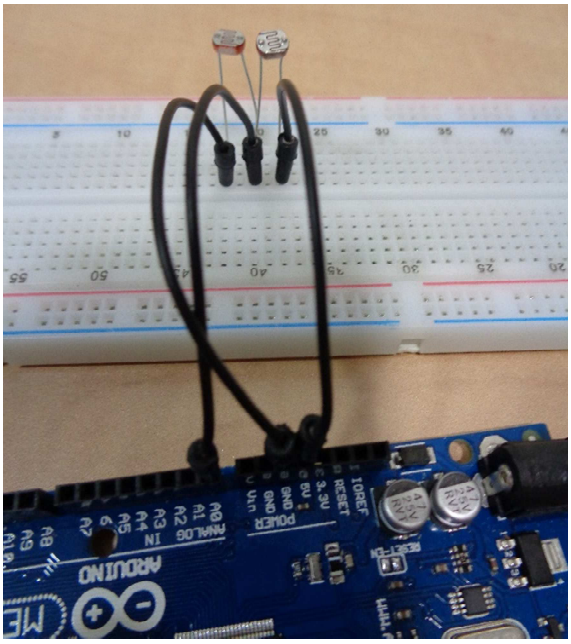
7.1. Programacions de proves prèvies

7.1.1. Programació de les proves de fotoresistències

Per tal de poder calibrar les fotoresistències LDR utilitzarem una Protoboard i muntarem l'esquema definitiu. Amb una mateixa incidència lumínica i igualant la variable de calibratge a cero comprovarem la diferencia de tensió donada entre fotoresistències utilitzant la comanda *Monitor Serie*.



Imatge 7.1. Connexionat per proves de LDR (Font pròpia)



Imatge 7.2. Detall del connexionat per proves de LDR (Font pròpia)

La diferencia amb el valor 2,5 V serà el valor que haurem de donar a la variable de calibratge del programa.

L'esquema del connexionat per la realització d'aquesta prova esta al Annex B2 d'aquest document.

En la programació, es comença definint el pin analògic al que es connectarà la senyal del LDR, una constant que s'utilitzarà per passar de bytes a volts i una variable per aconseguir que les dues fotoresistències a iguals valors de llum incident tinguin la mateixa caiguda de tensió.

Dins del void setup () només inicialitzem la comunicació serial.

A void loop () guardem la lectura del pin_LDR, el corregim tenint en compte el calibratge necessari entre les dues fotoresistències i convertim el valor a tensió (valors de 0-1023 a 0-5).

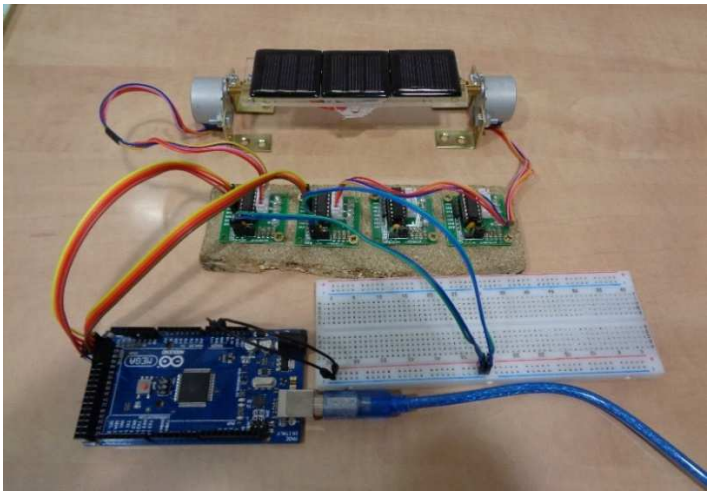
L'objectiu d'aquesta codi es veure la diferencia existent entre ambos LDR, i per tal d'equilibrar—ho, modificar el valor de la variable de calibratge; aconseguint uns valors en pantalla de 2.5 V. Aquest valor serà important per les següents proves.

```
// PROGRAMA DE CALIBRACIÓ LDR

// Defineix el pin A0 com al pin al que esta connectat el LDR
int pin_LDR = 0;

// Constant que converteix el valor sensat en valor de tensió
// 5 / 1023 = 0.0048875855

float kt = 0.0048875855;
int calibracio = 0;

void setup()
{
  // Inicialitza la comunicació serial a 9600 bits per segon
  Serial.begin(9600);
}


void loop()
{
  // Guarda en la variable valor_LDR la senyal de la tensió del
  // conjunt LDR
  float valor_LDR = (analogRead(pin_LDR) + calibracio) * kt;


  Serial.println(valor_LDR);
  delay(500);
}
```

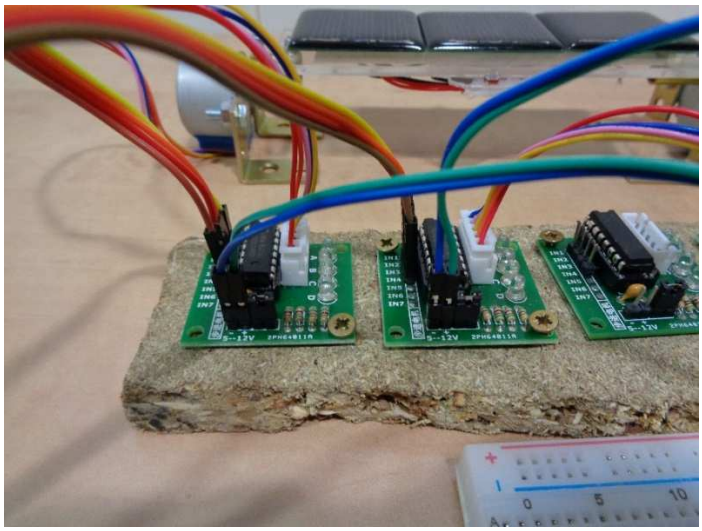
7.1.2. Prova de funcionament dels motors

L'objectiu d'aquesta prova de funcionament es corroborar que els motors funcionen, que giren cap on toca i veure si existeixen defectes d'alineament.

Durant el funcionament d'aquesta prova es corroborarà que ambos motors giren en sentits contraris (un a dretes i l'altre a esquerres). També que no existeixen problemes d'alineació d'eixos, es detecta al no cargolar les escaires a cap superfície i veure que en funcionament aquestes romanen estàtiques. En cas de que no fos així, s'afluixarien els cargols de les regletes fins aconseguir el alineament.



Imatge 7.3. Connexionat per proves de motors (Font pròpia)



Imatge 7.4. Detall del connexionat per proves de motors (Font pròpia)

L'esquema de connexionat es troba al annex B2 d'aquest document.

En la programació, es comença definint els pins digitals als que es connectaran les senyals cap a les plaques de control dels motor pas a pas i la seqüencia d'alimentació de les bobines; en aquest cas s'ha optat per una seqüencia de mitjos passos.

Dins de void setup () es configuren les senyals com a sortides, es realitza la comunicació sèrie i es realitza un quart de volta als motors. Aquest quart de volta es perquè tenint inicialment les plaques posicionades de forma totalment horitzontal amb el terra, aquestes es posin perpendiculars per tal que després al void loop () tots els moviments siguin de 180 graus cap a cada costat.

```
// PROGRAMA DE PROVES DE FUNCIONAMENT DELS MOTORS

// Defineix el pins als que estan connectats els motors

#define IN1A1  53
#define IN2A1  51
#define IN3A1  49
#define IN4A1  47

#define IN1A2  52
#define IN2A2  50
#define IN3A2  48
#define IN4A2  46

// Seqüencia de passos a seguir del tots els motors pas a pas
// En aquest cas es a mitjos passos (8 passos)

int Pas [8][4] =
{
  {1, 0, 0, 0},
  {1, 1, 0, 0},
  {0, 1, 0, 0},
  {0, 1, 1, 0},
  {0, 0, 1, 0},
  {0, 0, 1, 1},
  {0, 0, 0, 1},
  {1, 0, 0, 1}
};

void setup()
{

  // Inicialitza la comunicació serial a 9600 bits per segon
  Serial.begin(9600);

  // Es configuren tots els pins com a sortida
  pinMode(IN1A1, OUTPUT);
  pinMode(IN2A1, OUTPUT);
  pinMode(IN3A1, OUTPUT);
  pinMode(IN4A1, OUTPUT);

  pinMode(IN1A2, OUTPUT);
  pinMode(IN2A2, OUTPUT);
  pinMode(IN3A2, OUTPUT);
  pinMode(IN4A2, OUTPUT);
```

```
// Amb el receptor solar paral·lel al terra, fem un quart de volta

  for (int x = 0; x < 128; x++)
  {
    for (int i = 7; i > -1; i--)
    {
      digitalWrite(IN1A1, Pas[i][0]);
      digitalWrite(IN2A1, Pas[i][1]);
      digitalWrite(IN3A1, Pas[i][2]);
      digitalWrite(IN4A1, Pas[i][3]);

      digitalWrite(IN1A2, Pas[i][3]);
      digitalWrite(IN2A2, Pas[i][2]);
      digitalWrite(IN3A2, Pas[i][1]);
      digitalWrite(IN4A2, Pas[i][0]);
      delay(5);
    }
  }
}


// Realitzem mitja volta cap a cada costat. Dins del rang que es podria
// moure amb el hardware final

void loop()
{
  for (int x = 0; x < 256; x++)
  {
    for (int i = 0; i < 8; i++)
    {
      digitalWrite(IN1A1, Pas[i][0]);
      digitalWrite(IN2A1, Pas[i][1]);
      digitalWrite(IN3A1, Pas[i][2]);
      digitalWrite(IN4A1, Pas[i][3]);

      digitalWrite(IN1A2, Pas[i][3]);
      digitalWrite(IN2A2, Pas[i][2]);
      digitalWrite(IN3A2, Pas[i][1]);
      digitalWrite(IN4A2, Pas[i][0]);
      delay(5);
    }
  }
  for (int x = 0; x < 256; x++)
  {
    for (int i = 7; i > -1; i--)
    {
      digitalWrite(IN1A1, Pas[i][0]);
      digitalWrite(IN2A1, Pas[i][1]);
      digitalWrite(IN3A1, Pas[i][2]);
      digitalWrite(IN4A1, Pas[i][3]);

      digitalWrite(IN1A2, Pas[i][3]);
      digitalWrite(IN2A2, Pas[i][2]);
      digitalWrite(IN3A2, Pas[i][1]);
      digitalWrite(IN4A2, Pas[i][0]);
      delay(5);
```

```
      }
   }
}
```

7.2. Mètode 1. Cerca del punt de màxima tensió

Consisteix en censar la tensió donada per les plaques i comparar-la amb punts propers, fent que s'orienti la major part del temps allà on el valor donat es màxim.

En la programació, es comença definint els pins digitals als que es connectaran les senyals cap a les plaques de control dels motor pas a pas i la seqüencia d'alimentació de les bobines; en aquest cas s'ha optat per una seqüencia de mitjos passos. També es defineix una altre variable anomenada 'Dades' que s'utilitzarà a l'hora de fer la comunicació. En el pin A0 serà al que es connectarà la senyal provinent de les plaques solars.

Es declaren tres variables per guardar els valors de tensió de les cel·les solars en tres posicions diferents i una altre variable per tal de passar els valors de bytes a Volts (*kt)*.

Per tal de saber en tot moment en quin pas s'alimenten les bobines dels motors, es creen dues variables anomenades *' x '* i *' i ';* on *x* es el pas en el que esta alimentat el motor pas a pas i *i* son els passos anterior i posterior on es censa la tensió per tal de comparar-la amb la donada en la posició *x*.

Dins de void setup () es configuren les senyals com a sortides i es realitza la comunicació sèrie.

A void loop () guardem la lectura del pin_PV i convertim el valor a tensió (valors de 0-1023 a 0-5). Després fem avançar un pas als motors i es torna a guardar la lectura en una altre variable; depenent de si aquest últim es major, igual o menor; es realitzarà unes comandes especifiques per tal de posicionar-se en el punt de màxima tensió.

```
// PROGRAMA DE SEGUIDOR VMAX

// Definim els pins on tenim connectades les bobines

#define IN1A1  53
#define IN2A1  51
#define IN3A1  49
#define IN4A1  47

#define IN1A2  52
#define IN2A2  50
#define IN3A2  48
#define IN4A2  46

char Dades = (';');
```

```
// Definim el pin on es connectara la senyal de les cel·les
fotovoltaiques

int pin_PV = 0;

// Declarem unes variables que actuará com a variable temporal, on
// emmagatzemarem els valors de tensió de les plaques solars

int valor_PV = 0;
int valor_PV_1 = 0;
int valor_PV_2 = 0;

// Constant que converteix el valor sensat en valor de tensió
// 5 / 1023 = 0.0048875855

float kt = 0.0048875855;


// Secuencia de pasos a seguir del tots els motors pas a pas
// En aquest cas es a mitjos pasos (8 pasos)

int Pas [8][4] =
{
  {1, 0, 0, 0},
  {1, 1, 0, 0},
  {0, 1, 0, 0},
  {0, 1, 1, 0},
  {0, 0, 1, 0},
  {0, 0, 1, 1},
  {0, 0, 0, 1},
  {1, 0, 0, 1}
};

// Creem dues variables per a cada eix; on 'x' es el pas en el que es
troba
// i 'i' es el pas en el que analitza la tensió

int x = 0;
int i = 0;

void setup()
{
  // inicializa la comunicación serial en 9600 bits por segundo
  Serial.begin(9600);

  // Todos los pines en modo salida
  pinMode(IN1A1, OUTPUT);
  pinMode(IN2A1, OUTPUT);
  pinMode(IN3A1, OUTPUT);
  pinMode(IN4A1, OUTPUT);

  pinMode(IN1A2, OUTPUT);
  pinMode(IN2A2, OUTPUT);
  pinMode(IN3A2, OUTPUT);
  pinMode(IN4A2, OUTPUT);

}
```

```
void loop()
{
  int valor_PV = (analogRead(pin_PV)) * kt;

  i = x + 1;

  if (i > 7)
  {
    i = 0;
  }

  digitalWrite(IN1A1, Pas[i][0]);
  digitalWrite(IN2A1, Pas[i][1]);
  digitalWrite(IN3A1, Pas[i][2]);
  digitalWrite(IN4A1, Pas[i][3]);

  digitalWrite(IN1A2, Pas[i][3]);
  digitalWrite(IN2A2, Pas[i][2]);
  digitalWrite(IN3A2, Pas[i][1]);
  digitalWrite(IN4A2, Pas[i][0]);

  delay(50);

  int valor_PV_1 = (analogRead(pin_PV) * kt);

  if (valor_PV_1 > valor_PV)
  {
    x = i;

    delay (15);

    Serial.print(x);
    Serial.print(Dades);
    Serial.print(valor_PV_1);
    Serial.print(Dades);
    Serial.println(millis());
    delay(300);
  }

  else if (valor_PV_1 < valor_PV)
  {
    digitalWrite(IN1A1, Pas[x][0]);
    digitalWrite(IN2A1, Pas[x][1]);
    digitalWrite(IN3A1, Pas[x][2]);
    digitalWrite(IN4A1, Pas[x][3]);

    digitalWrite(IN1A2, Pas[x][3]);
    digitalWrite(IN2A2, Pas[x][2]);
    digitalWrite(IN3A2, Pas[x][1]);
    digitalWrite(IN4A2, Pas[x][0]);

    i = x - 1;

    if (i < 0)
    {
      i = 7;
    }
```

```
    delay(50);

    digitalWrite(IN1A1, Pas[i][0]);
    digitalWrite(IN2A1, Pas[i][1]);
    digitalWrite(IN3A1, Pas[i][2]);
    digitalWrite(IN4A1, Pas[i][3]);

    digitalWrite(IN1A2, Pas[i][3]);
    digitalWrite(IN2A2, Pas[i][2]);
    digitalWrite(IN3A2, Pas[i][1]);
    digitalWrite(IN4A2, Pas[i][0]);

    delay(50);

    int valor_PV_2 = (analogRead(pin_PV) * kt);
    x = i;

    Serial.print(x);
    Serial.print(Dades);
    Serial.print(valor_PV_2);
    Serial.print(Dades);
    Serial.println(millis());

    delay(300);
  }
  else
  {
    digitalWrite(IN1A1, Pas[x][0]);
    digitalWrite(IN2A1, Pas[x][1]);
    digitalWrite(IN3A1, Pas[x][2]);
    digitalWrite(IN4A1, Pas[x][3]);

    digitalWrite(IN1A2, Pas[x][3]);
    digitalWrite(IN2A2, Pas[x][2]);
    digitalWrite(IN3A2, Pas[x][1]);
    digitalWrite(IN4A2, Pas[x][0]);

    delay(1000);
  }
}
```

L'esquema de connexionat es troba al annex B2 d'aquest document.

7.3. Mètode 2. Seguidor mitjançant la posició solar

La programació d'aquest mètode de seguiment es massa complex com per poder fer-ho amb un Arduino Mega, faria falta un hardware on la comunicació serial amb una base de dades fos més senzill.

7.4. Mètode 3. Seguidor mitjançant fotoresistències

Aquest mètode de seguidor, en funció de la llum rebuda per cada LDR, fa variar la orientació de la superfície receptora per tal de que la caiguda de tensió de cada fotoresistència sigui igual.

En la programació, es comença definint els pins digitals als que es connectaran les senyals cap a les plaques de control dels motor pas a pas i la seqüencia d'alimentació de les bobines; en aquest cas s'ha optat per una seqüencia de mitjos passos. Es defineix una variable anomenada 'Dades' que s'utilitzarà a l'hora de fer la comunicació.

Els pins A0 i A1 seran als que es connectaran les senyals provinents de les plaques solars i de les fotoresistències, respectivament. Aquests valors seran emmagatzemats en dos variables creades per aquesta finalitat; valor_PV i valor_LDR.

Dins de void setup () es configuren les senyals com a sortides i es realitza la comunicació sèrie.

A void loop () guardem les lectures de pin_PV i pin_LDR, i els convertim en valor de tensió (valors de 0-1023 a 0-5). En funció de la magnitud de valor_LDR; si es superior o inferior a 2,5 V, el motor gira cap un sentit o l'altre. Això significa que s'ha de tenir cura a l'hora d'alimentar els LDR i al fer les connexions del motor, ja que sinó no s'aconseguiria que funcionés correctament.

L'esquema de connexionat es troba al annex B2 d'aquest document.

```
//PROGRAMA SEGUIDOR LDR

// Definim els pins on es connectaran les bobines dels motors pas a pas
// on cada motor tindrà un sentit de gir.

#define IN1A1  53
#define IN2A1  51
#define IN3A1  49
#define IN4A1  47

#define IN1A2  52
#define IN2A2  50
#define IN3A2  48
#define IN4A2  46

char Dades = (';');

// Definim el pin on es connectarà la senyal de les cel·les
fotovoltaiques

int pin_PV = 0;

// Declarem un coma flotant que actuarà com a variable temporal, on
// emmagatzemarem els valors de tensió de les plaques solars
```

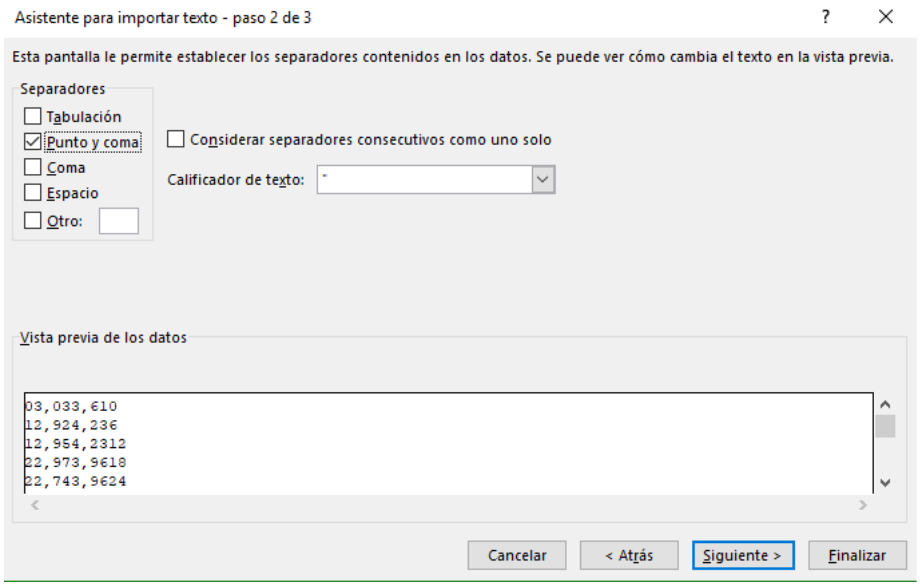
```
float valor_PV = 0;

// Definim el pin on es connectarà la senyal dels LDR

int pin_LDR = 1;

// Creem una variable per tal d'equilibrar els LDR de cada eix
// Aquest valor serà determinat per la prova realitzada anteriorment

int calibracio = 0;

// Declarem un coma flotant que actuarà com a variable temporal, on
// emmagatzemarem els valors de tensió absoluts

float valor_LDR = 0;

// Seqüencia de passos a seguir del tots els motors pas a pas
// En aquest cas es a mitjos passos (8 passos)

int Pas [8][4] =
{
  {1, 0, 0, 0},
  {1, 1, 0, 0},
  {0, 1, 0, 0},
  {0, 1, 1, 0},
  {0, 0, 1, 0},
  {0, 0, 1, 1},
  {0, 0, 0, 1},
  {1, 0, 0, 1}
};

// Creem una variable 'i' que es el pas en el que analitza la tensió dels
LDR

int i = 0;

void setup()
{
  // Inicialitza la comunicació serial a 9600 bits per segon
  Serial.begin(9600);

  // Es configuren tots els pins com a sortida
  pinMode(IN1A1, OUTPUT);
  pinMode(IN2A1, OUTPUT);
  pinMode(IN3A1, OUTPUT);
  pinMode(IN4A1, OUTPUT);

  pinMode(IN1A2, OUTPUT);
  pinMode(IN2A2, OUTPUT);
  pinMode(IN3A2, OUTPUT);
  pinMode(IN4A2, OUTPUT);

}

void loop()
{
```

```
// Es fa una lectura del valor de la tensió diferencial entre les
// dues fotoresistències i de la tensió donada per les plaques.

float valor_LDR = analogRead(pin_LDR) * (0.0048828125) + calibracio;
float valor_PV = analogRead(pin_PV) * (0.0048828125);

Serial.print(i);
Serial.print(Dades);
Serial.print(valor_LDR);
Serial.print(Dades);
Serial.print(valor_PV);
Serial.print(Dades);
Serial.println(millis());


// Depenent si el valors diferencials entre fotoresistències es positiu
// o negatiu, els motors pas a pas giraran en un sentit o un altre


if (valor_LDR > 2.52)
{
  i = i + 1;

  // Ens assegurem de que al passar del pas 7 torni a començar al pas 0
  // i a al revés; fent així que la matriu estigui tancada

  if (i > 7)
  {
    i = 0;
  }

  digitalWrite(IN1A1, Pas[i][0]);
  digitalWrite(IN2A1, Pas[i][1]);
  digitalWrite(IN3A1, Pas[i][2]);
  digitalWrite(IN4A1, Pas[i][3]);

  digitalWrite(IN1A2, Pas[i][3]);
  digitalWrite(IN2A2, Pas[i][2]);
  digitalWrite(IN3A2, Pas[i][1]);
  digitalWrite(IN4A2, Pas[i][0]);

  delay(5);

  float valor_LDR = analogRead(pin_LDR) * (0.0048828125) + calibracio;
  float valor_PV = analogRead(pin_PV) * (0.0048828125);

  Serial.print(i);
  Serial.print(Dades);
  Serial.print(valor_LDR);
  Serial.print(Dades);
  Serial.print(valor_PV);
  Serial.print(Dades);
  Serial.println(millis());

}

delay(5);
```

```
  if (valor_LDR < 2.48)

  {
    i = i - 1;

    // Ens assegurem de que al passar del pas 7 torni a començar al pas 0
    // i al revés; fent així que la matriu estigui tancada

    if (i < 0)
    {
      i = 7;
    }

    digitalWrite(IN1A1, Pas[i][0]);
    digitalWrite(IN2A1, Pas[i][1]);
    digitalWrite(IN3A1, Pas[i][2]);
    digitalWrite(IN4A1, Pas[i][3]);

    digitalWrite(IN1A2, Pas[i][3]);
    digitalWrite(IN2A2, Pas[i][2]);
    digitalWrite(IN3A2, Pas[i][1]);
    digitalWrite(IN4A2, Pas[i][0]);

    delay(5);

    float valor_LDR = analogRead(pin_LDR) * (0.0048828125) + calibracio;
    float valor_PV = analogRead(pin_PV) * (0.0048828125);

    Serial.print(i);
    Serial.print(Dades);
    Serial.print(valor_LDR);
    Serial.print(Dades);
    Serial.print(valor_PV);
    Serial.print(Dades);
    Serial.println(millis());
  }
}
```

8. Resultats obtinguts

Les dades donades per cada codi es poden veure utilitzant la opció de Monitor Serie del Arduino IDE. Les dades es poden extreure de varies formes, però la utilitzada ha sigut copiant el llistat de dades donades i guardant-les en un bloc de notes. Aquest document posteriorment s'ha de obrir amb un Excel i marcar la casella de que els punts i comes passen a ser els separadors de columnes.



Imatge 8.1. Exportar resultats a excel (Font pròpia)

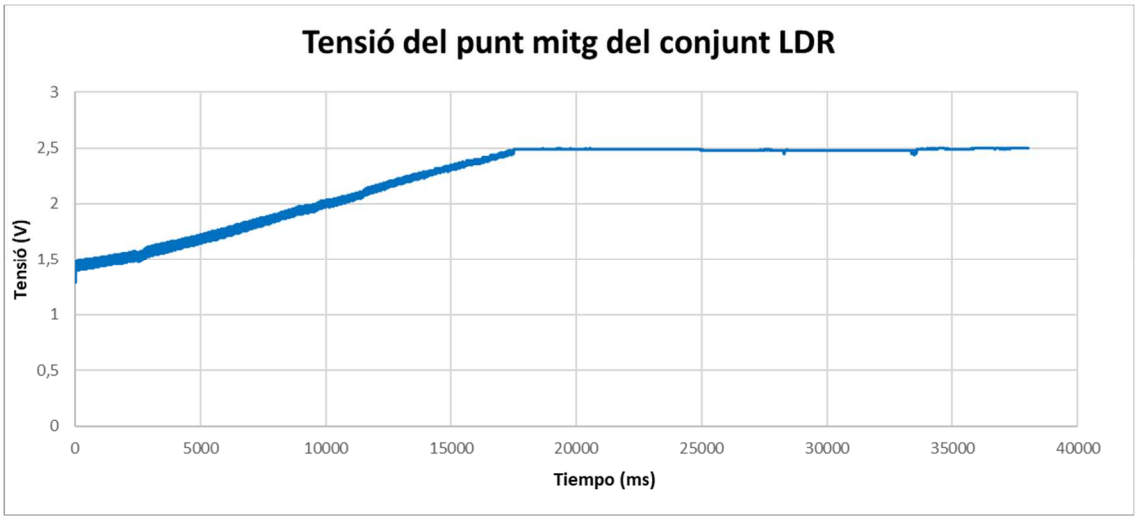
8.1. Seguidor mitjançant fotoresistències

Per aquest tipus de seguidor, s'han realitzat tres diferents proves; llum artificial i llum natural fins estabilitzar-se i una altre durant més de vint minuts per tan de corroborar que segueix amb precisió els moviments Solars.

8.1.1. Seguidor de llum artificial

S'ha utilitzat el codi de Arduino IDE seguidor amb LDR. Realitzat a la nit, per tal d'evitar gran part de interferències solars possibles durant 35 segons; temps suficient en donar per bo els resultats obtinguts i veure la tendència de la tensió donada per el punt mig dels LDR.

El llistat de dades generades es troben en el disc CD ja que es un gran numero de files de dades.

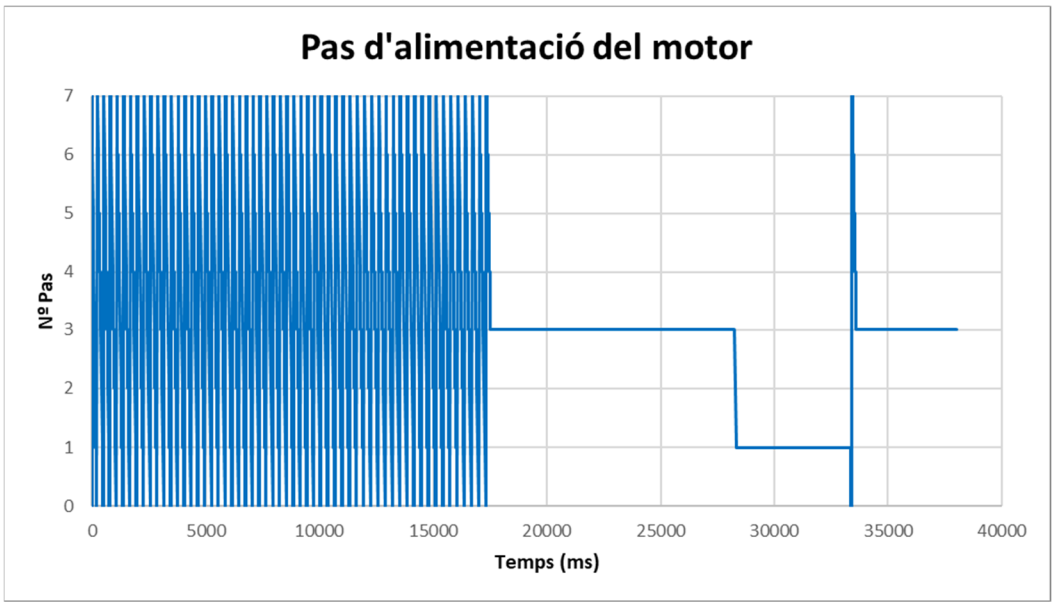


Gràfica 8.1. Tensió donada per el conjunt LDR (Font pròpia)

La tensió del punt mig dels LDR ha de ser en regim estacional el més proper a 2,5 V, aquest es el valor en que les dos fotoresistències reben la mateixa quantitat de llum; i per tant, la superfície receptora està perpendicular al objecte generador de llum.

En la Gràfica 8.1 el valor de tensió al principi de l'assaig es proper a 1,5 V, i durant 17 segons va augmentant fins arribar al valor desitjat de 2,5 V.

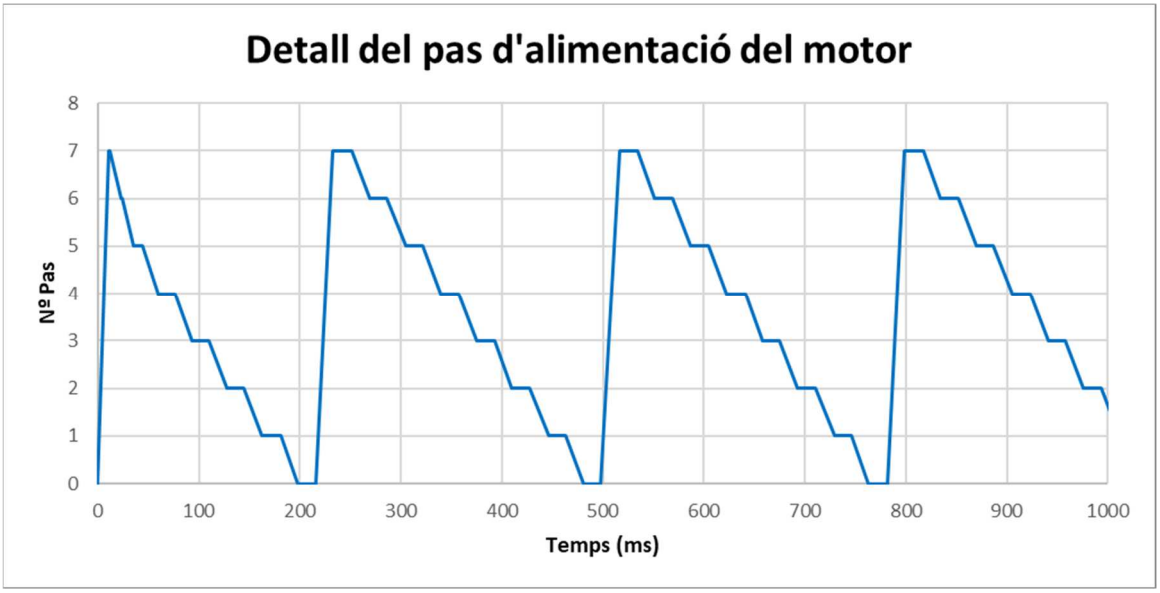
Aquest es manté fins la finalització de la prova. Es poden apreciar als 28.000 ms i als 33.000 ms que apareix una pertorbació; això es degut a que en aquests dos punts s'ha perdut la perpendicularitat amb la font lumínica o ha hagut alguna interferència entre la font i un dels sensors.



Gràfica 8.2. Passos d'alimentació dels motors (Font pròpia)

La Gràfica 8.2 tracta del moviment dels passos en funció del temps; aquests queden aturats en el pas tercer als 17 segons, que coincideix amb moment on la tensió es igual a 2,5 V.

En els segons 28 i 33 aproximadament, es produeixen variacions en els passos on també coincideixen amb la Gràfica 8.1. on la tensió no era 2,5 V, i en la Gràfica 8.2. es pot veure que el sistema rectifica la posició d'orientació per tal de continuar obtenint la tensió objectiu.

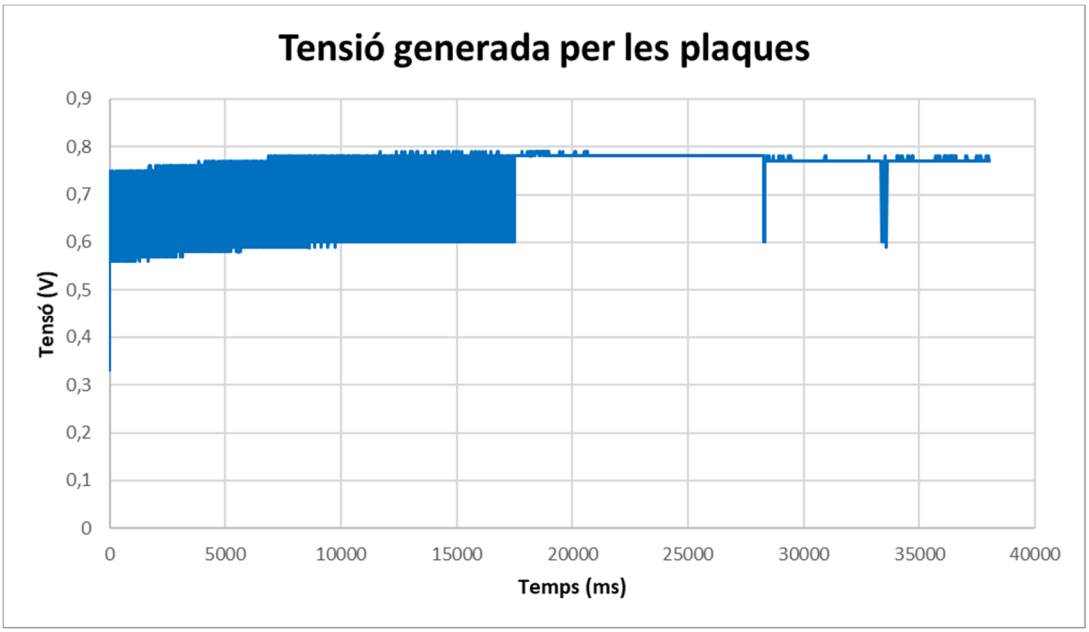


Gràfica 8.3. Detall dels passos (Font pròpia)

El temps que triga en fer una volta completa a la matriu de senyals de les bobines es de 200 ms. Aquest temps es podria fer més curt reduint els temps dels *delay* de la programació utilitzada. No existeixen moments on els motors girin cap a la direcció equivocada, en tot moment giren en el sentit adequat. Això es perquè sabent si la tensió dels LDR es major o menor del valor consigna, cap on ha de girar; no cal censar la tensió donada en altres punts per poder determinar el sentit correcte.

La tensió generada per les plaques es mínima, degut a que aquesta prova es va realitzar durant la nit, els valors son deguts a possibles filtracions i reflexes de llum natural (lluna,...) i independentment de la tensió donada, el hardware s'orienta a la perfecció amb la llum natural.

Aquesta tensió s'estabilitza en el mateix moment en que el hardware es queda aturat i també apareixen variacions en els mateixos instats que en les altres gràfiques.



Gràfica 8.4. Tensió donada per les plaques (Font pròpia)

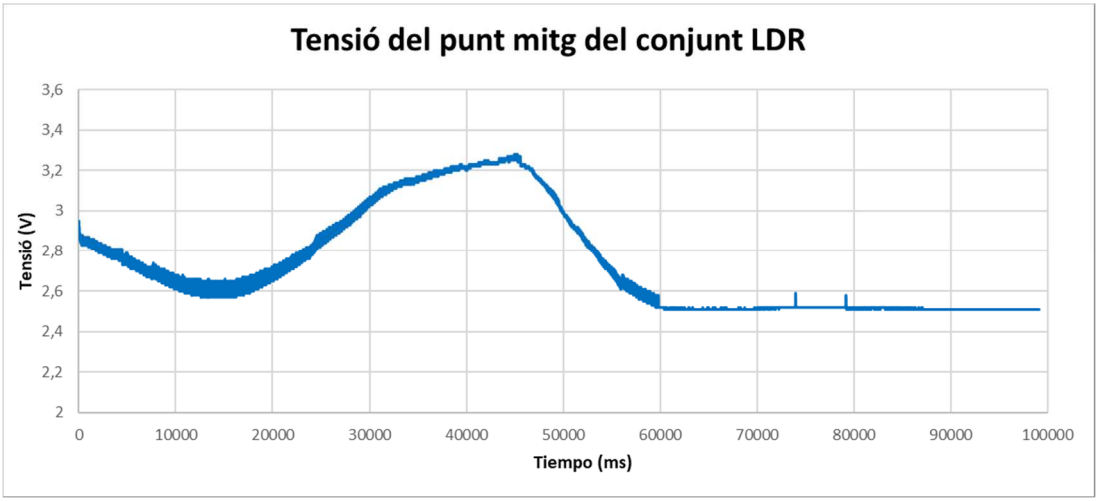
Es conclou que el seguidor d'un eix mitjançant sensors LDR amb llum artificial funciona correctament i que els resultats son els esperats.

8.1.2. Seguidor de llum solar

S'ha utilitzat el codi de Arduino IDE seguidor amb LDR. En aquest cas es provarà el seguidor LDR amb llum natural per tal de veure com respon i si al arribar als 2,5 V, la tensió generada per les plaques, es màxima.

Realitzat durant el dia, amb una incidència solar directa durant 100 segons fins s'ha comprovat que ha arribat a regim permanent.

El llistat de dades generades es troben en el disc CD ja que es un gran numero de files de dades.



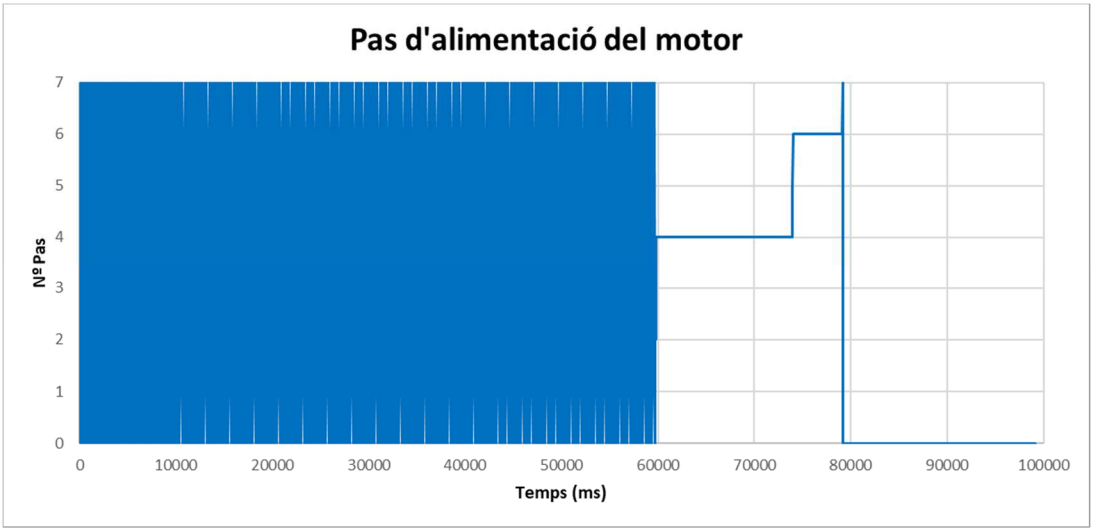
Gràfica 8.5. Tensió donada per el conjunt LDR (Font pròpia)

La tensió del punt mig dels LDR ha de ser en regim estacional el més proper a 2,5 V, aquest es el valor en que les dos fotoresistències reben la mateixa quantitat de llum; i per tant, la superfície receptora està perpendicular al objecte generador de llum.

En la Gràfica 8.5 el valor de tensió al principi de l'assaig es proper a 2,9 V, i durant 17 segons va variant fins estabilitzar-se al cap d'un minut a un voltatge de 2,5 V. Posteriorment a aquest instant apareixen algunes irregularitats però continua proper al valor objectiu.

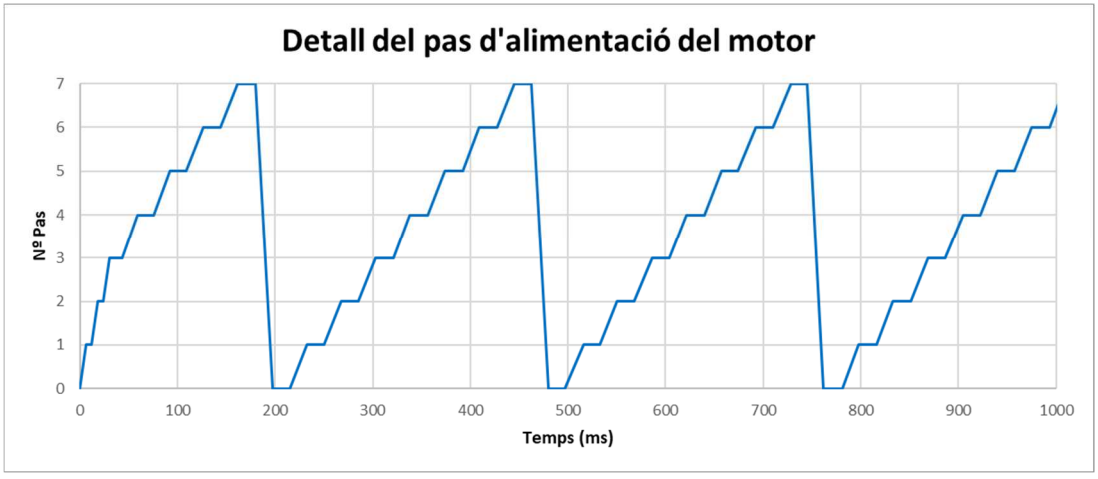
La forma d'ona que apareix al principi es degut a que inicialment la llum solar no incideix directament sobre cap LDR però un d'ells rep més irradiància que l'altre (el més proper al Sol). Segons es va rectificant la posició del hardware per tal l'orientar-se perpendicularment, el LDR més proper al Sol rep directament la llum i genera una diferencia major de caiguda de tensió i això provoca aquesta tipus de senyal.

Com ha passat amb el seguidor LDR amb llum artificial, apareixen uns pics de tensió als 74.000 ms i als 79.000 ms. Aquests son deguts a que per el moviment solar, la tensió a caiguda de tensió de cada LDR no es la mateixa; i per tant, ha de fer un pas i tornar a estar dins del valor objectiu.



Gràfica 8.6. Passos d'alimentació dels motors (Font pròpia)

Els motors han estat girant fins als 60 segons, que es quan s'ha aconseguit tenir els 2,5 V de senyal als LDR. Als 74 segons els motors han realitzat 2 passos, i uns altres 2 als 79 segons. Aquests tornen a coincidir amb les variacions de tensió vists a la Gràfica 8.5.

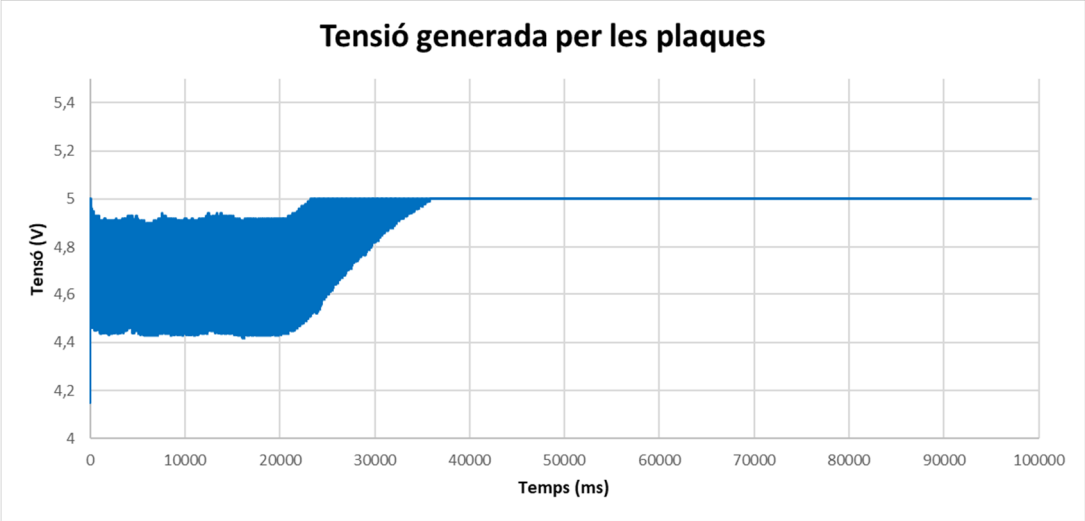


Gràfica 8.7. Detall dels passos (Font pròpia)

El temps que triga en fer una volta completa a la matriu de senyals de les bobines es de 200 ms, coincideix amb el temps del seguidor LDR amb llum artificial. Aquest temps es podria fer més curt reduint els temps dels *delay* de la programació utilitzada. No existeixen moments on els motors girin cap a la direcció equivocada, en tot moment giren en el sentit adequat. Això es perquè sabent si la tensió dels LDR es major o menor del valor consigna, cap on ha de girar; no cal censar la tensió donada en altres punts per poder determinar el sentit correcte.

La tensió generada per les plaques inicialment ja es alta, es degut a que l'assaig ha sigut realitzat a mitja tarda, i que sense tenir les cel·les solars perpendiculars als rajos solars, absorbeixen gran part de radiació difusa i reflectida.

En la Gràfica 8.7., fins als 30 segons el valor de la tensió es transitori i després s'estabilitza als 5 V. Les entrades d'Arduino només llegeixen fins als 5 V. El valor de la tensió de les plaques amb la dels LDR estable podria arribar als 6 V o més.



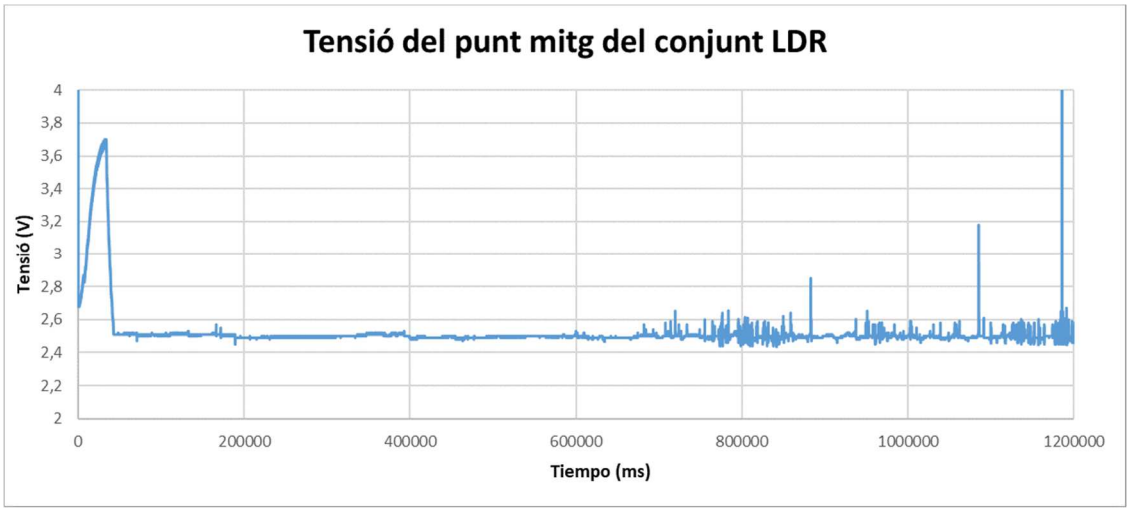
Gràfica 8.8. Tensió donada per les plaques (Font pròpia)

Es conclou que el seguidor d'un eix mitjançant sensors LDR amb llum natural s'orienta correctament.

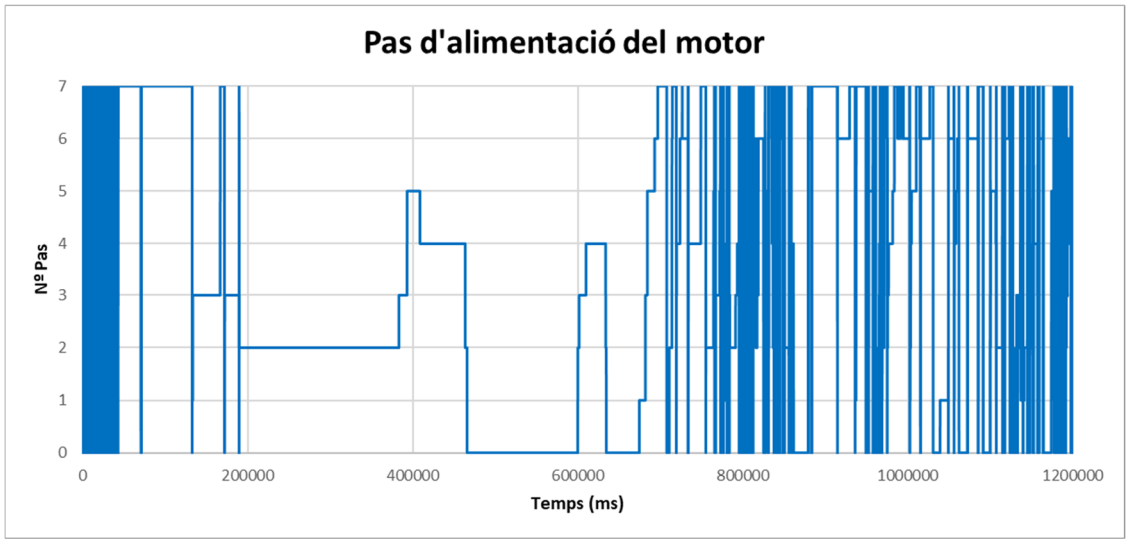
Els valors de tensió generats per les plaques solars no eren els esperats. La tensió es molt superior a la donada per el fabricant. Amb les tres plaques utilitzades la tensió màxima esperada eren 4,5 V. La explicació d'aquest fet es que els valors del fabricant son per una irradiància i temperatura concreta, i que per altres valors la tensió pot variar.

8.1.3. Seguidor de llum solar 20 minuts

S'ha realitzat la prova anterior amb un temps de 20 minuts una tarda solejada per analitzar la resposta del hardware i software amb llarg temps.



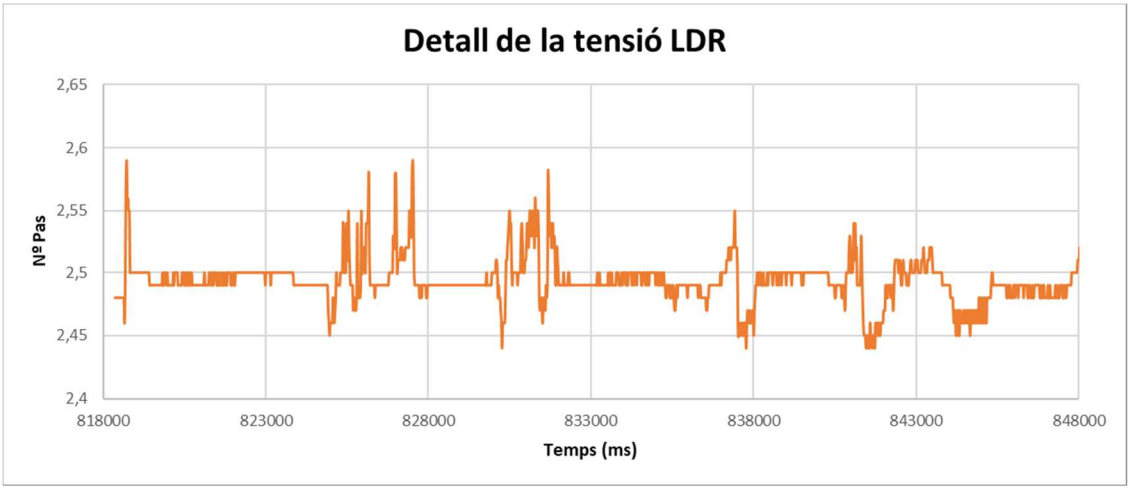
Gràfica 8.9. Tensió donada per el conjunt LDR (Font pròpia)



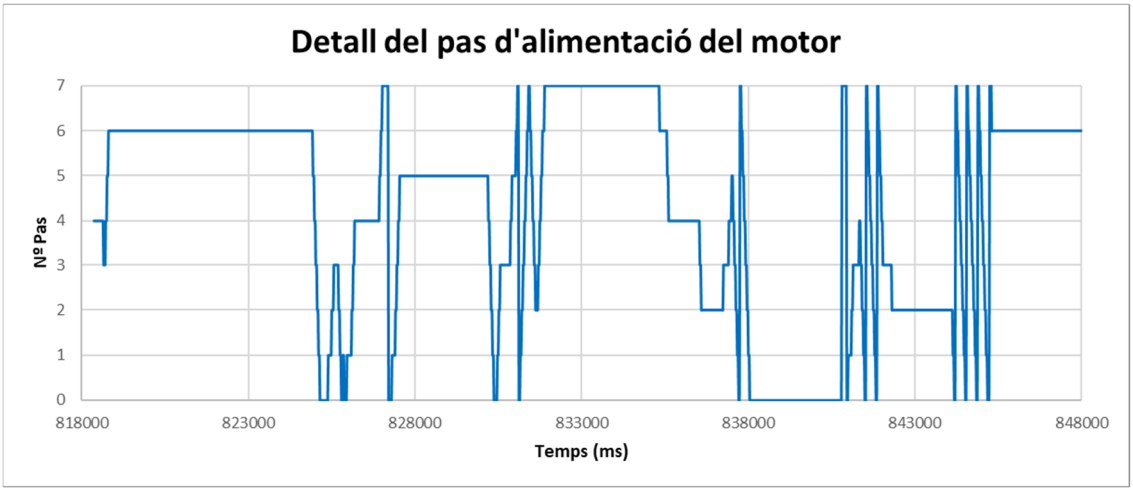
Gràfica 8.10. Passos d'alimentació dels motors (Font pròpia)

Les gràfiques 8.9. i 8.10 il·lustren la tensió i els passos durant els 20 minuts d'experiment. En menys d'un minut el sistema ha entrat en regim estacionari i des de llavors s'ha mantingut proper al valor de 2,5 V. Existeixen alguns pics de tensió però ràpidament son corregits alineant-se de nou amb la perpendicular de la radiació solar.

En regim permanent, tota variació de la tensió te una resposta en el sentit de gir dels motors tal que fa que aquesta primera torni a valors inicials.



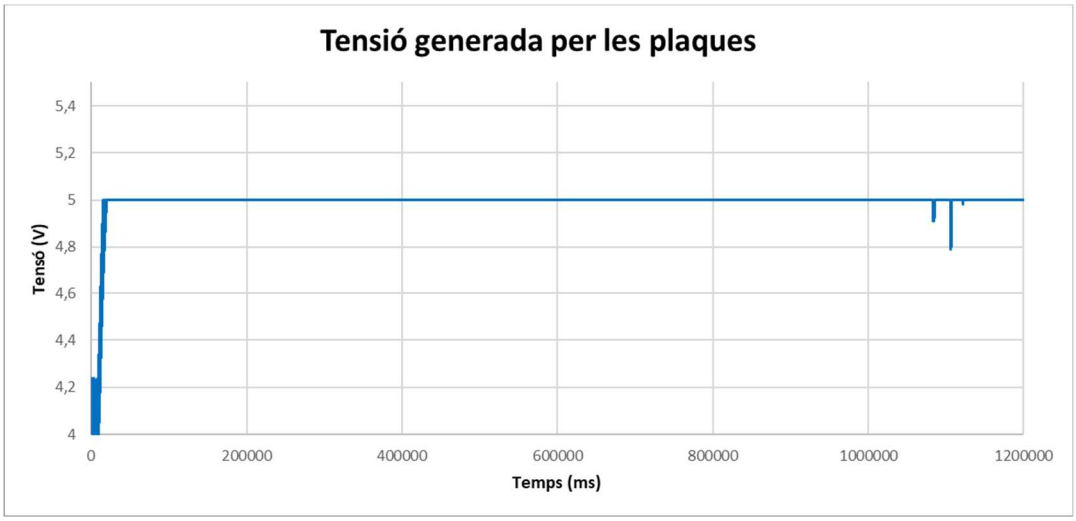
Gràfica 8.11. Detall tensió donada per el conjunt LDR (Font pròpia)



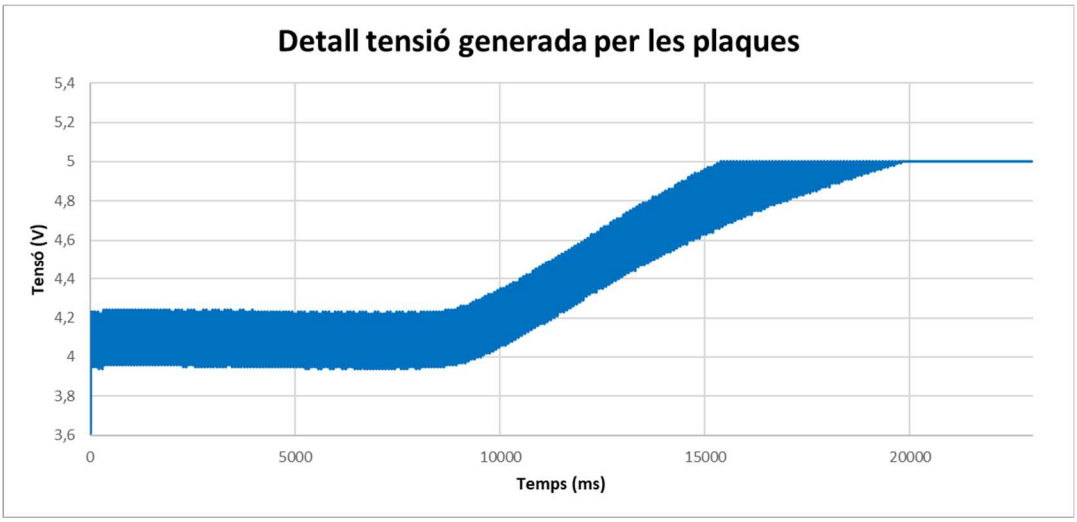
Gràfica 8.12. Detall dels passos (Font pròpia)

Fent un detall a un interval de temps concret, es pot veure com per a cada augment de tensió respecte el valor 2,5 V; existeix un altre augment de pas donat per els motors.

El temps de resposta del software a aquestes variacions son molt rapides, de l'ordre de pocs mil·lisegons, fet que facilita la precisió de seguiment.



Gràfica 8.13 Tensió donada per les plaques (Font pròpia)



Gràfica 8.14. Detall tensió donada per les plaques (Font pròpia)

La tensió donada per les plaques es manté constant al llarg de tot el experiment, exceptuant un parell de cops al final d'aquest. Aquesta tensió arriba al seu valor màxim als 20 segons.

S'ha de tenir el compte que Arduino només llegeix fins als 5 V i que per tant a partir del segon 15 ja podríem tenir valors superiors a aquests.

Pas	valor_LDR	valor_PV	millis()
7	2,5	5	719217
7	2,5	5	719239
7	2,5	5	719259
7	2,5	5	719280
7	2,51	5	719301
7	2,5	5	719322
7	2,51	5	719342
7	2,52	5	719364
0	2,57	5	719384
0	2,54	5	719405
1	2,54	5	719426
1	2,52	5	719447
1	2,52	5	719467
2	2,54	5	719489
2	2,65	5	719509
3	2,6	5	719529
3	2,5	5	719550
3	2,51	5	719571
3	2,51	5	719592
3	2,51	5	719612
3	2,52	5	719634
3	2,52	5	719654
3	2,52	5	719675
3	2,51	5	719696

Taula 8.15. Dades extretes de Arduino IDE (Font pròpia)

La taula 8.15, es un extracte de dades corresponents a un petit interval de temps de l'experiment del seguidor LDR amb llum natural durant 20 minuts.

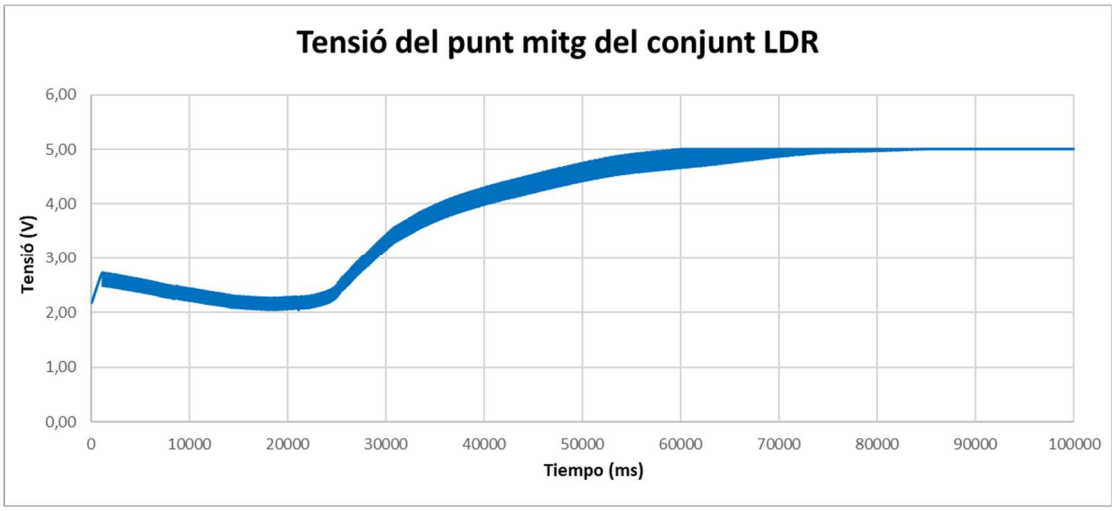
Es veu que quan la tensió supera el valor de 2,52 V, el valor dels passos augmenta també, fent reduir la tensió del conjunt de fotoresistències.

Es conclou que el seguidor d'un eix mitjançant sensors LDR amb llum natural s'orienta correctament a llarg termini. Tot i aparèixer en moments donats pics de tensió superiors e inferiors a 2,5 V, el hardware es capaç de modificar la seva posició de manera idònia per tal d'adaptar-se a un medi canviant i poder mantenir la superfície receptora perpendicular als raigs solars.

8.2. Seguidor mitjançant la tensió generada per les cel·les solars

S'ha utilitzat el codi de Arduino IDE seguidor de tensió màxima. Realitzat durant el dia, amb una incidència solar directa durant 100 segons fins s'ha comprovat que ha arribat a regim permanent.

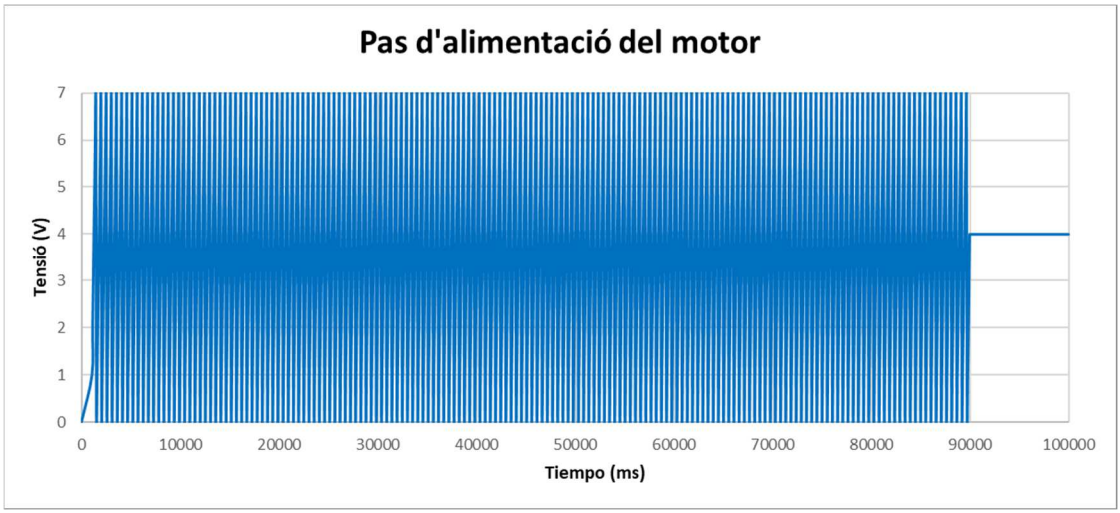
El llistat de dades generades es troben en el disc CD ja que es un gran numero de files de dades.



Gràfica 8.16. Tensió generada per les plaques (Font pròpia)

La Gràfica 8.16. apareix la tensió generada per cada instant de temps, la qual s'acaba estabilitzant a 5 V, recordem que aquest valor es el màxim que pot llegir una entrada de Arduino.

El temps en arribar a regim estacionari es major que utilitzant les fotoresistències però per contrapartida la precisió donada per aquesta es major (ho seria en el cas que poguéssim registrar valors superiors a 5 V també).



Gràfica 8.17. Passos d'alimentació dels motors (Font pròpia)

Com hem vist amb els altres assajos, el motor deixa de girar quan la tensió registrada es màxima. Tot i no tenir les lectures reals màximes generades per les plaques, corroborem que la programació funciona correctament i els resultats son els esperats.

Conclusions

Tenint com a objectiu un hardware de baix cost, s'han proposat una sèrie d'articles per composar el seguidor solar. S'ha aconseguit tenir una estructura operativa i funcional preparada per utilitzar-la per poder carregar diferents codis i continuar aprofundint en els seguidors solars.

Vistos els resultats, la conclusió es que tot i no haver aconseguit mesurar amb Arduino la tensió màxima generada per les plaques s'ha pogut assegurar de que tant el hardware i la programació funcionen correctament.

Aquest projecte no està acabat, continuant amb les proves al hardware d'un eix, es pot buscar el punt de màxima potencia; per fer això seria necessari un inversor al que censar la intensitat i tensió donades a la sortida. També es implementar una pantalla LCD per poder llegir els valors en temps directe sense necessitat d'obrir el Arduino IDE o un convertidor per tal de que independentment de la tensió generada aquesta sempre sigui constant (útil per alimentar dispositius mòbils i el propi Arduino).

En quant als seguidors de dos eixos, encara hi ha bastant per fer, des de el dimensionament el segon eix i verificar que funciona correctament fins a poder controlar grans superfícies de plaques solars destinades a la producció d'energia elèctrica.

També es pot implementar en diversos projectes ja existents.

Pressupost i/o Anàlisi Econòmica

	€ / UD	UD	€
28BYJ-48 + ULN2003A	3,18	2	6,36
CEL·LES SOLARS	3,75	3	11,25
PLACA DE METACRILAT 3MM – A4	3,75	1	3,75
ESCAIRES BRICOMATADES 20MM	0,25	4	1,00
KIT ARDUINO MEGA	23,37	1	23,37
REGLETA 16 MM2	3,55	1	3,55
PETIT MATERIAL	5,36	1	5,36
TOTAL			54,64

Taula A.1 Pressupost de materials

El cost total del hardware i software utilitzat en aquest projecte es de **54,64 €**.

La partida dels motors pas a pas 28BYJ-48 i la placa de control ULN2003A inclou el cablejat suficient com per poder fer les connexions necessàries per fer-los funcionar correctament. El kit Arduino Mega; a més a més d'incloure la placa Arduino i el cable USB, porta sensors, resistències i actuadors de diversos tipus. El petit material entra cablejat extra, fil de estany, silicona, cargols, femelles, etc...

La valoració de les hores es la següent:

	€ / H	H	€
DISSENY I SELECCIÓ HARDWARE	20	65	1.300
PROGRAMACIÓ	20	80	1.600
CONSTRUCCIÓ HARDWARE	20	100	2.000
PROVES	20	35	700
TOTAL		280	5.600

Taula A.1 Pressupost de mà d'obra

El total de la realització d'aquest projecte puja a **5.654,64 €**

CONDICIONS GENERALS

- Validesa de l'oferta: 60 dies.
- No està inclòs cap treball no especificat en aquesta oferta
- No està inclòs el 21% d'IVA.
- Els treballs a realitzar que no estiguin previstos en el pressupost, es cobraran a part.
- Termini de lliurament: Consultar

Bibliografia

1. Aplicación online gratuita para el cálculo instalación solar fotovoltaica. A: [en línia]. Disponible a: http://calculationsolar.com/.

2. Arduino. *Index @ Www.Arduino.Cc* [en línia]. 2014. 2014. Disponible a: http://www.arduino.cc/.

3. Ayushi, M. i Ingole, N. Arduino based solar tracking system. A: . 2016, p. 61-66.

4. Banerjee, R. Solar Tracking System. A: *International Journal of Scientific and Research Publications* [en línia]. 2014, Vol. 5, núm. 1, p. 2250-3153. [Consulta: 3 maig 2018]. Disponible a: www.ijsrp.org.

5. El Motaleb, D.. A.M.A. Seguimiento del punto de máxima potencia en Sistemas Fotovoltaicos. A: [en línia]. 2010, núm. August 2016, p. 1-18. DOI 10.13140/RG.2.1.1003.4964. Disponible a: http://bibing.us.es/proyectos/abreproy/70172/fichero/Resumen.pdf.

6. Fortanet, J.G. Gira-sol fotovoltaic: Seguidor solar automàtic per a la captació de l"energia del sol. A: [en línia]. 2016, Disponible a: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/53381/8/jfortanetTFG0716memòria.pdf.

7. MENDOZA GUEVARA, A. Seguidor dinámico solar para incrementar la eficiencia de placas fotovoltaicas mediante fotocélulas y servomotores controlados por un microcontrolador. A: [en línia]. 2015, p. 113. Disponible a: https://riunet.upv.es/bitstream/handle/10251/58621/MENDOZA - Seguidor dinámico solar para incrementar la eficiencia de placas fotovoltaicas mediante....pdf?sequence=3.

8. Rosario, S. Monitorización en tiempo real de seguidores solares fotovoltaicos en doble eje. A: [en línia]. 2016, Disponible a: https://uvadoc.uva.es/bitstream/10324/17049/1/TFG-P-361.pdf.

9. CIRCUITO ULN2003 – Electrónica: teoría y práctica. A: [en línia]. Disponible a: http://electronica-teoriaypractica.com/circuito-uln2003/.

10. FADISEL. A: [en línia].. Disponible a: https://fadisel.com/es/.

11. LOGO! - El Futuro de la Industria - Siemens. A: [en línia]. Disponible a: https://w5.siemens.com/spain/web/es/industry/automatizacion/simatic/controladores_modulares/logo/pages/default.aspx.

12. x10 fotorresistencia 5mm GL5528 sensor fotorresistor LDR luminosidad luz DIY para Arduino. A: [en línia]. Disponible a: http://inven.es/componentes-electronicos/43-fotoresistencia-5mm-gl5528.html.

13. Ardumanía. A: [en línia]. Disponible a: http://www.ardumania.es/.

14. Raspberry Pi 3 Model B - Raspberry Pi. A: [en línia]. Disponible a: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

15. PIC24FJ256DA206-I/PT - COM-11700 - SparkFun Electronics. A: [en línia]..Disponible a: https://www.sparkfun.com/products/11700.

16. Energía Renovable Peru con Deltavolt. A: [en línia]. Disponible a: http://deltavolt.pe/.

17. El ULN2003. driver de salida para microcontroladores | Inventable.eu. A: [en línia]. Disponible a: https://www.inventable.eu/2018/02/09/uln2003-driver-salida-microcontroladores/.

18. Motor paso a paso ¿que es y como funciona? - Ingeniería mecafenix. A: [en línia].. Disponible a: http://www.ingmecafenix.com/electricidad-industrial/motor-paso-a-paso/.

19. Fritzing Fritzing. A: [en línia]. Disponible a: http://fritzing.org/home/.

20. Seguidor Solar Con Motores a Paso + Arduino. A: [en línia]. Disponible a: http://www.instructables.com/id/Stepper-Motor-Solar-Tracker-Seguidor-Solar-con-mot/.

21. Pagina www.areatecnologia.com. A: [en línia]. Disponible a: http://www.areatecnologia.com.

22. Obtener certificado energético. Certificado de eficiencia energética. A: [en línia]. Disponible a: http://www.certificadosenergeticos.com/.

Annex A: Documentació tècnica

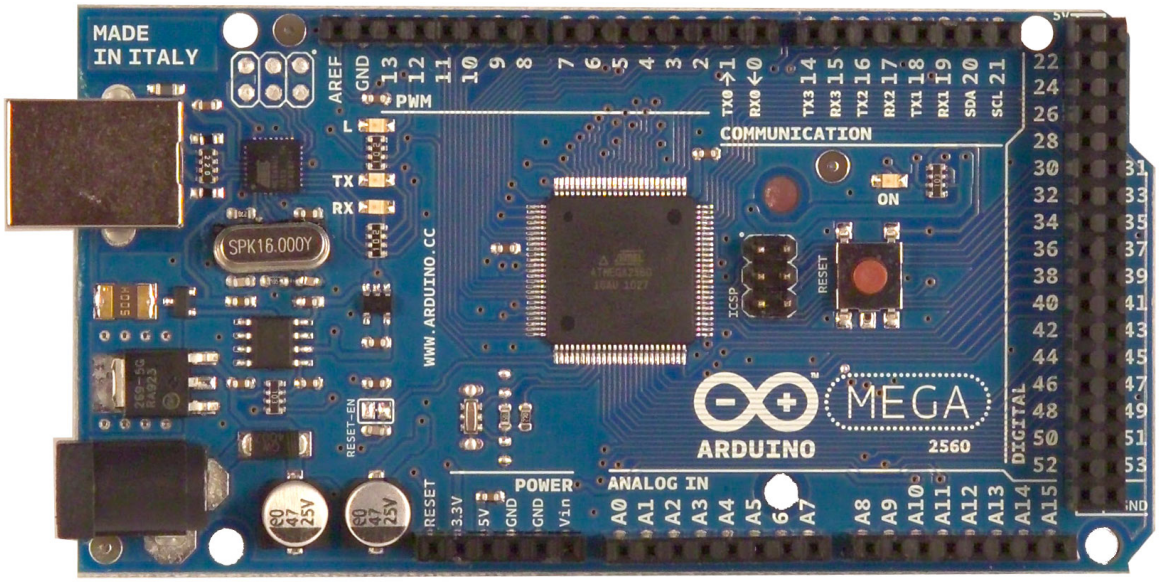
A1. Fulls de característiques

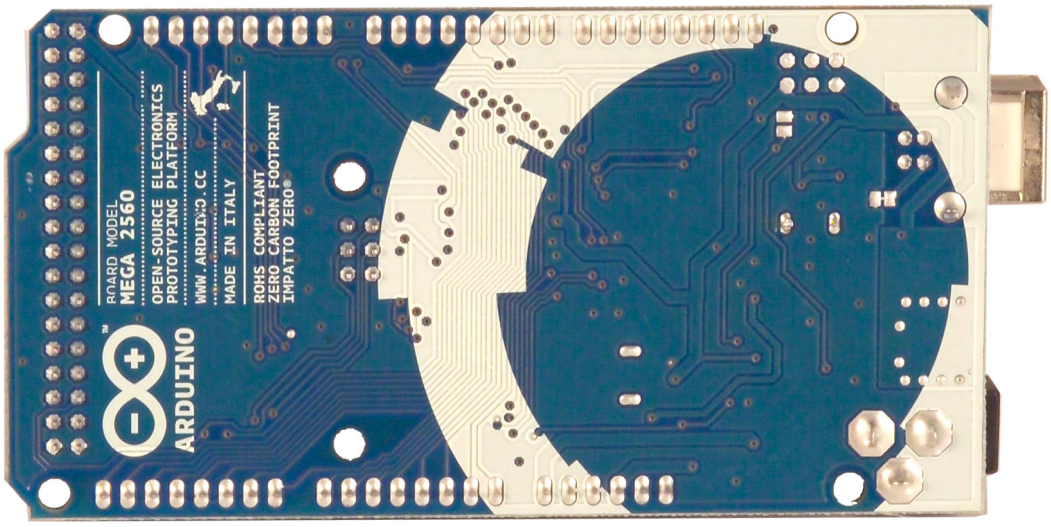
www.robotshop.com

La robotique à votre service! - Robotics at your service!

ARDUINO

Arduino Mega 2560 Datasheet





Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Schematic & Reference Design

EAGLE files: arduino-mega2560-reference-design.zip

Schematic: [arduino-mega2560-schematic.pdf]

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Power

The Arduino Mega can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode() , digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH

value, the LED is on, when the pin is LOW, it's off.
- **I2C: 20 (SDA) and 21 (SCL).** Support I2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I2C pins on the Duemilanove or Diecimila.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).
A SoftwareSerial library allows for serial communication on any of the Mega2560's digital pins.
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. For SPI communication, use the SPI library.

Programming

The Arduino Mega can be programmed with the Arduino software (download). For details, see the reference and tutorials.
The ATmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It

communicates using the original STK500 protocol (reference, C header files).
You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.
This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.
The Mega2560 contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

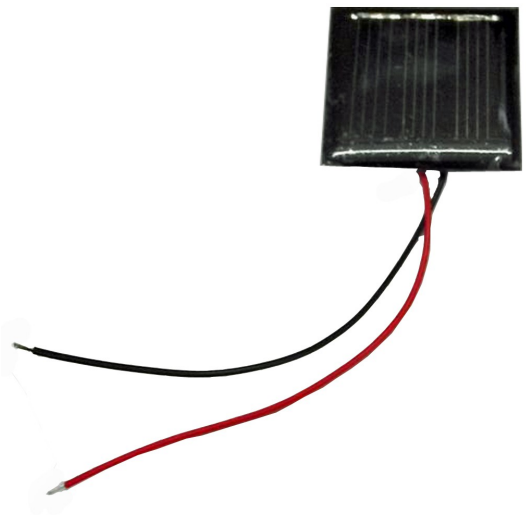
The Arduino Mega2560 has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega2560 PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega2560 is designed to be compatible with most shields designed for the Uno, Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega2560 and Duemilanove / Diecimila. *Please note that I^2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).*

SOLAR CELL
PANNEAU SOLAIRE
PANEL SOLAR
1.5 V 150 mW
C-0138



TECHNICAL CHARACTERISTICS

Power.....150 mW
Voltage Voc.....1,5 V. DC
Isc.....100 mA.
Vm.....1,0 V.
Im.....75 mA.
Measures.....40 x 40 mm
Connection.....AWG26
Lot.....4 units

Photovoltaic solar cells and high-performance miniaturized. They are ideal for classroom practices of technology, electricity, electronics, crafts, and robotics for any type of installation that requires a very small cell size and high performance. See our catalog the various special solar engine can be driven directly by these cells .

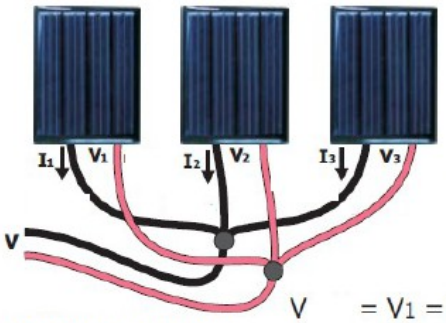
Mounting and installation. For cell fixation is recommended to use double sided tape on the back. Preferably a spongy base tape. The cell should stand we face the direct sunlight. Its performance depends on the light received. It can run on the inside, if the cell is illuminated with an incandescent lamp, preferably halogen. Not suitable for fluorescent lighting or compact fluorescent lamps .

Connection. Photovoltaic cells can be grouped into assemblies "series", "Parallel" and "mixed." When connecting two or more identical cells in series, the resulting voltage is the sum of all of them and the current intensity is the same for all. When connecting two or more identical cells in parallel, the voltage will be the same for all, with the resulting current intensity equal to the sum of all intensities. With serial, parallel or mixed is possible to obtain voltage and current we require. It is very important to respect the polarity shown in the diagrams .

Les panneaux solaires photovoltaïques de haute performance miniaturisés. Elles sont idéales pour les pratiques en classe de technologie, de l'électricité, l'électronique, de l'artisanat, la robotique et de montage pour tout type de cellule nécessaire par une très petite taille et haute performance. Voir nos spéciaux moteurs solaires en nôtre catalogue qui peuvent être entraîné directement par ces panneaux solaires.
Pour fixer le panneau recommandé d'utiliser un adhésif double face sur le dos. De préférence, une bande de base spongieux. Le panneau doit être placé adressée directement au soleil. Sa performance dépend de l'éclairement reçu. Vous pouvez travailler à l'intérieur, si le panneau les lumières une lampe à incandescence, halogènes de préférence. Ne convient pas pour l'éclairage fluorescent ou des lampes fluorescentes compactes.
Les panneaux photovoltaïques peuvent être regroupés en ensembles "de série", "parallèle" et "mixtes". Connexion de deux ou plusieurs panneaux, en nombre égal, la tension qui en résulte est la somme de tous et l'intensité du courant est la même pour tous. Connexion de deux ou plusieurs panneaux en parallèle égale, la tension sera la même pour tous, avec l'intensité de courant résultant égale à la somme de toutes les intensités. Avec série, parallèle ou mixte est possible d'obtenir la tension et le courant nous avons besoin. Il est très important de respecter la polarité indiquée dans les diagrammes .

Células solares fotovoltaicas miniaturizadas y de alto rendimiento. Son ideales para prácticas en el aula de tecnología, electricidad, electrónica, manualidades, robótica y para cualquier tipo de montaje que precise una célula de tamaño muy reducido y altas prestaciones.
Consulte en nuestro catálogo los diversos motores solares especiales que pueden ser accionados directamente por estas células.
Montaje e instalación. Para la fijación de la célula se recomienda usar cinta adhesiva de doble cara en el dorso. Preferiblemente una cinta con base esponjosa.
La célula debe situarse encarada a los rayos solares directos. Su rendimiento depende de la iluminación recibida.

Puede funcionar en el interior, si se ilumina la célula con una lámpara de incandescencia, preferiblemente halógena. No es adecuada para iluminación de tubos fluorescentes o lámparas fluorescentes compactas.
Conexión. Las células fotovoltaicas pueden agruparse en montajes “serie”, “Paralelo” y “mixto”.
Al conectar dos o más células iguales en serie, la tensión resultante será la suma de todas ellas y la intensidad de la corriente será la misma para todas.
Al conectar dos o más células iguales en paralelo, la tensión será la misma para todas, siendo la intensidad de la corriente resultante igual a la suma de todas las intensidades.
Mediante conexiones serie, paralelo o mixtas es posible obtener las tensión y corriente que precisemos.
Es muy importante respectar la polaridad que se indica en los esquemas.

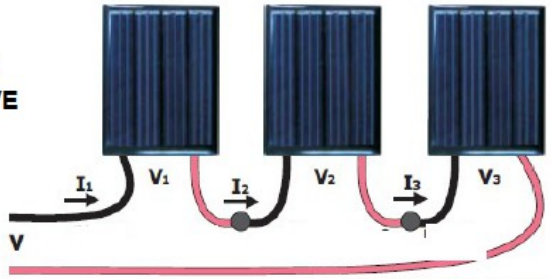


Parallel connection

$V = V_1 = V_2 = V_3$

$I = I_1 + I_2 + I_3$

RED = POSITIVE
BLACK = NEGATIVE



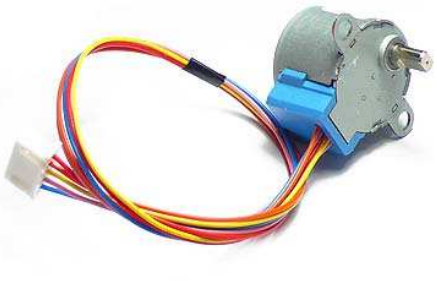
Series connection

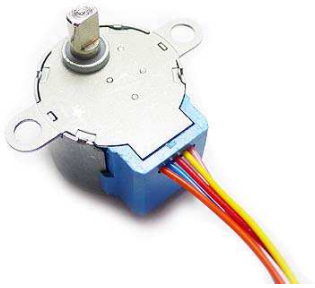
$V = V_1 + V_2 + V_3$

$I = I_1 = I_2 = I_3$

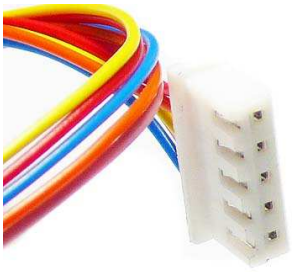
28BYJ-48 – 5V Stepper Motor

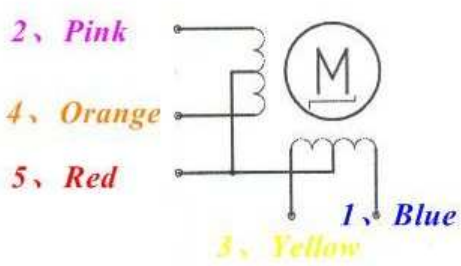
The 28BYJ-48 is a small stepper motor suitable for a large range of applications.

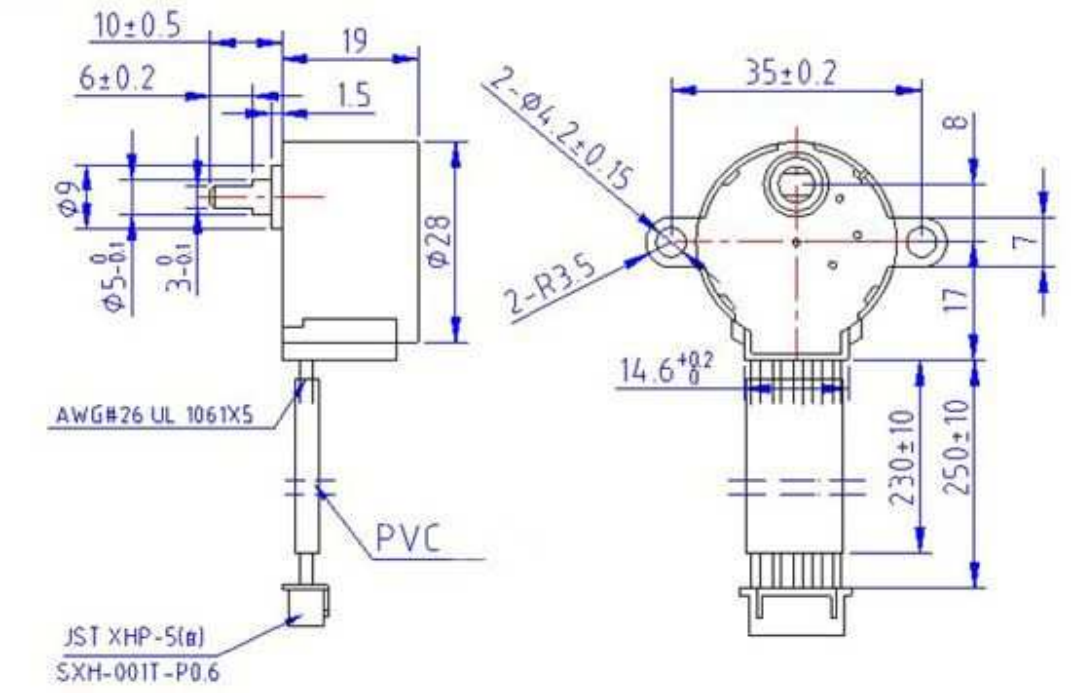




Rated voltage ：	5VDC
Number of Phase	4
Speed Variation Ratio	1/64
Stride Angle	5.625°/64
Frequency	100Hz
DC resistance	50Ω±7%(25℃)
Idle In-traction Frequency	> 600Hz
Idle Out-traction Frequency	> 1000Hz
In-traction Torque	>34.3mN.m(120Hz)
Self-positioning Torque	>34.3mN.m
Friction torque	600-1200 gf.cm
Pull in torque	300 gf.cm
Insulated resistance	>10MΩ(500V)
Insulated electricity power	600VAC/1mA/1s
Insulation grade	A
Rise in Temperature	<40K(120Hz)
Noise	<35dB(120Hz,No load,10cm)
Model	28BYJ-48 – 5V







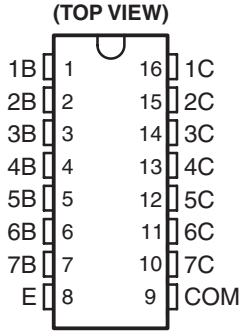
HIGH-VOLTAGE, HIGH-CURRENT DARLINGTON TRANSISTOR ARRAYS

Check for Samples: ULN2002A, ULN2003A, ULN2003AI, ULN2004A, ULQ2003A, ULQ2004A

FEATURES

- **500-mA-Rated Collector Current (Single Output)**
- **High-Voltage Outputs: 50 V**
- **Output Clamp Diodes**
- **Inputs Compatible With Various Types of Logic**
- **Relay-Driver Applications**

ULN2002A . . . N PACKAGE
ULN2003A . . . D, N, NS, OR PW PACKAGE
ULN2004A . . . D, N, OR NS PACKAGE
ULQ2003A, ULQ2004A . . . D OR N PACKAGE
(TOP VIEW)



DESCRIPTION

The ULN2002A, ULN2003A, ULN2003AI, ULN2004A, ULQ2003A, and ULQ2004A are high-voltage high-current Darlington transistor arrays. Each consists of seven npn Darlington pairs that feature high-voltage outputs with common-cathode clamp diodes for switching inductive loads. The collector-current rating of a single Darlington pair is 500 mA. The Darlington pairs can be paralleled for higher current capability. Applications include relay drivers, hammer drivers, lamp drivers, display drivers (LED and gas discharge), line drivers, and logic buffers. For 100-V (otherwise interchangeable) versions of the ULN2003A and ULN2004A, see the SN75468 and SN75469, respectively.

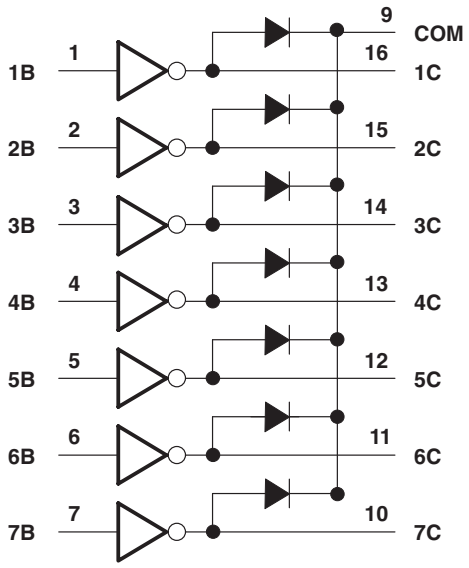
The ULN2001A is a general-purpose array and can be used with TTL and CMOS technologies. The ULN2002A is designed specifically for use with 14-V to 25-V PMOS devices. Each input of this device has a Zener diode and resistor in series to control the input current to a safe limit. The ULN2003A and ULQ2003A have a 2.7-kΩ series base resistor for each Darlington pair for operation directly with TTL or 5-V CMOS devices. The ULN2004A and ULQ2004A have a 10.5-kΩ series base resistor to allow operation directly from CMOS devices that use supply voltages of 6 V to 15 V. The required input current of the ULN/ULQ2004A is below that of the ULN/ULQ2003A, and the required voltage is less than that required by the ULN2002A.

ORDERING INFORMATION(1)

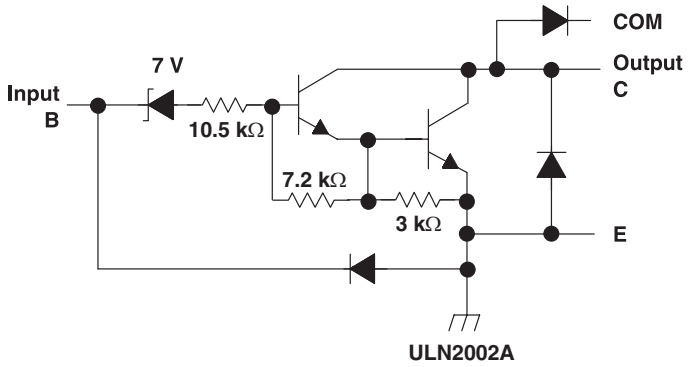
T_A	PACKAGE(2)		ORDERABLE PART NUMBER	TOP-SIDE MARKING
–20°C to 70°C	PDIP – N	Tube of 25	ULN2002AN	ULN2002AN
			ULN2003AN	ULN2003AN
			ULN2004AN	ULN2004AN
	SOIC – D	Tube of 40	ULN2003AD	ULN2003A
		Reel of 2500	ULN2003ADR	
		Reel of 2500	ULN2003ADRG3	
		Tube of 40	ULN2004AD	ULN2004A
		Reel of 2500	ULN2004ADRG3	
	SOP – NS	Reel of 2000	ULN2003ANSR	ULN2003A
			ULN2004ANSR	ULN2004A
	TSSOP – PW	Tube of 90	ULN2003APW	UN2003A
		Reel of 2000	ULN2003APWR	
–40°C to 85°C	PDIP – N	Tube of 25	ULQ2003AN	ULQ2003A
			ULQ2004AN	ULQ2004AN
	SOIC – D	Tube of 40	ULQ2003AD	ULQ2003A
		Reel of 2500	ULQ2003ADR	
		Tube of 40	ULQ2004AD	ULQ2004A
		Reel of 2500	ULQ2004ADR	
	SOP – NS	Reel of 2000	ULN2003AINSR	ULN2003AI
–40°C to 105°C	PDIP – N	Tube of 425	ULN2003AIN	ULN2003AIN
	SOIC – D	Tube of 40	ULN2003AID	ULN2003AI
		Reel of 2500	ULN2003AIDR	
	TSSOP – PW	Reel of 2500	ULN2003AIPWR	UN2003AI

(1) For the most current package and ordering information, see the Package Option Addendum at the end of this document, or see the TI web site at www.ti.com.
(2) Package drawings, thermal data, and symbolization are available at www.ti.com/packaging.

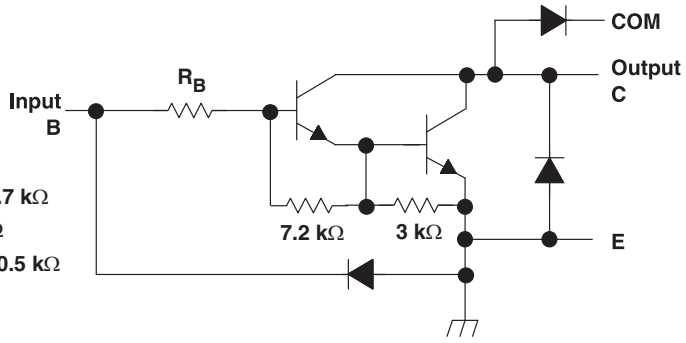
LOGIC DIAGRAM



SCHEMATICS (EACH DARLINGTON PAIR)



ULN2002A



ULN/ULQ2003A: R_B = 2.7 kΩ
ULN2003AI: R_B = 2.7 kΩ
ULN/ULQ2004A: R_B = 10.5 kΩ

ULN2003A, ULN2003AI, ULN2004A, ULQ2003A, ULQ2004A

All resistor values shown are nominal.

The collector-emitter diode is a parasitic structure and should not be used to conduct current. If the collector(s) go below ground an external Schottky diode should be added to clamp negative undershoots.

ABSOLUTE MAXIMUM RATINGS(1)

at 25°C free-air temperature (unless otherwise noted)

			MIN	MAX	UNIT
V_{CC}	Collector-emitter voltage			50	V
	Clamp diode reverse voltage(2)			50	V
V_I	Input voltage(2)			30	V
	Peak collector current	See Figure 14 and Figure 15		500	mA
I_{OK}	Output clamp current			500	mA
	Total emitter-terminal current			–2.5	A
T_A	Operating free-air temperature range	ULN200xA	–20	70	°C
		ULN200xAI	–40	105	
		ULQ200xA	–40	85	
		ULQ200xAT	–40	105	
θ_{JA}	Package thermal impedance(3) (4)	D package		73	°C/W
		N package		67	
		NS package		64	
		PW package		108	
θ_{JC}	Package thermal impedance(5) (6)	D package		36	
		N package		54	
T_J	Operating virtual junction temperature			150	°C
	Lead temperature for 1.6 mm (1/16 inch) from case for 10 seconds			260	°C
T_{stg}	Storage temperature range		–65	150	°C

(1) Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) All voltage values are with respect to the emitter/substrate terminal E, unless otherwise noted.
(3) Maximum power dissipation is a function of $T_J(max)$, θ_{JA}, and T_A. The maximum allowable power dissipation at any allowable ambient temperature is $P_D = (T_J(max) – T_A)/\theta_{JA}$. Operating at the absolute maximum T_J of 150°C can affect reliability.
(4) The package thermal impedance is calculated in accordance with JESD 51-7.
(5) Maximum power dissipation is a function of $T_J(max)$, θ_{JC}, and T_A. The maximum allowable power dissipation at any allowable ambient temperature is $P_D = (T_J(max) – T_A)/\theta_{JC}$. Operating at the absolute maximum T_J of 150°C can affect reliability.
(6) The package thermal impedance is calculated in accordance with MIL-STD-883.

ELECTRICAL CHARACTERISTICS

T_A = 25°C

PARAMETER		TEST FIGURE	TEST CONDITIONS		ULN2002A			UNIT
					MIN	TYP	MAX	
$V_{I(on)}$	On-state input voltage	Figure 6	V_{CE} = 2 V,	I_C = 300 mA			13	V
$V_{CE(sat)}$	Collector-emitter saturation voltage	Figure 4	I_I = 250 µA,	I_C = 100 mA		0.9	1.1	V
			I_I = 350 µA,	I_C = 200 mA		1	1.3	
			I_I = 500 µA,	I_C = 350 mA		1.2	1.6	
V_F	Clamp forward voltage	Figure 7	I_F = 350 mA			1.7	2	V
I_{CEX}	Collector cutoff current	Figure 1	V_{CE} = 50 V,	I_I = 0			50	µA
		Figure 2	V_{CE} = 50 V, T_A = 70°C	I_I = 0			100	
				V_I = 6 V			500	
$I_{I(off)}$	Off-state input current	Figure 2	V_{CE} = 50 V,	I_C = 500 µA	50	65		µA
I_I	Input current	Figure 3	V_I = 17 V			0.82	1.25	mA
I_R	Clamp reverse current	Figure 6	V_R = 50 V	T_A = 70°C			100	µA
							50	
C_i	Input capacitance		V_I = 0,	f = 1 MHz			25	pF

ELECTRICAL CHARACTERISTICS

T_A = 25°C

PARAMETER		TEST FIGURE	TEST CONDITIONS		ULN2003A			ULN2004A			UNIT
					MIN	TYP	MAX	MIN	TYP	MAX	
$V_{I(on)}$	On-state input voltage	Figure 6	V_{CE} = 2 V	I_C = 125 mA						5	V
				I_C = 200 mA			2.4			6	
				I_C = 250 mA			2.7				
				I_C = 275 mA						7	
				I_C = 300 mA			3				
				I_C = 350 mA						8	
$V_{CE(sat)}$	Collector-emitter saturation voltage	Figure 5	I_I = 250 µA,	I_C = 100 mA		0.9	1.1		0.9	1.1	V
			I_I = 350 µA,	I_C = 200 mA		1	1.3		1	1.3	
			I_I = 500 µA,	I_C = 350 mA		1.2	1.6		1.2	1.6	
I_{CEX}	Collector cutoff current	Figure 1	V_{CE} = 50 V,	I_I = 0			50			50	µA
		Figure 2	V_{CE} = 50 V, T_A = 70°C	I_I = 0			100			100	
				V_I = 6 V						500	
V_F	Clamp forward voltage	Figure 8	I_F = 350 mA			1.7	2		1.7	2	V
$I_{I(off)}$	Off-state input current	Figure 3	V_{CE} = 50 V, T_A = 70°C,	I_C = 500 µA	50	65		50	65		µA
I_I	Input current	Figure 4	V_I = 3.85 V			0.93	1.35				mA
			V_I = 5 V						0.35	0.5	
			V_I = 12 V						1	1.45	
I_R	Clamp reverse current	Figure 7	V_R = 50 V				50			50	µA
				T_A = 70°C			100			100	
C_i	Input capacitance		V_I = 0,	f = 1 MHz		15	25		15	25	pF

ELECTRICAL CHARACTERISTICS

T_A = 25°C

PARAMETER		TEST FIGURE	TEST CONDITIONS		ULN2003AI			UNIT
					MIN	TYP	MAX	
$V_{I(on)}$	On-state input voltage	Figure 6	V_{CE} = 2 V	I_C = 200 mA			2.4	V
				I_C = 250 mA			2.7	
				I_C = 300 mA			3	
$V_{CE(sat)}$	Collector-emitter saturation voltage	Figure 5	I_I = 250 µA,	I_C = 100 mA		0.9	1.1	V
			I_I = 350 µA,	I_C = 200 mA		1	1.3	
			I_I = 500 µA,	I_C = 350 mA		1.2	1.6	
I_{CEX}	Collector cutoff current	Figure 1	V_{CE} = 50 V,	I_I = 0			50	µA
V_F	Clamp forward voltage	Figure 8	I_F = 350 mA			1.7	2	V
$I_{I(off)}$	Off-state input current	Figure 3	V_{CE} = 50 V,	I_C = 500 µA	50	65		µA
I_I	Input current	Figure 4	V_I = 3.85 V			0.93	1.35	mA
I_R	Clamp reverse current	Figure 7	V_R = 50 V				50	µA
C_i	Input capacitance		V_I = 0,	f = 1 MHz		15	25	pF

ELECTRICAL CHARACTERISTICS

T_A = –40°C to 105°C

PARAMETER		TEST FIGURE	TEST CONDITIONS		ULN2003AI			UNIT
					MIN	TYP	MAX	
$V_{I(on)}$	On-state input voltage	Figure 6	V_{CE} = 2 V	I_C = 200 mA			2.7	V
				I_C = 250 mA			2.9	
				I_C = 300 mA			3	
$V_{CE(sat)}$	Collector-emitter saturation voltage	Figure 5	I_I = 250 µA,	I_C = 100 mA		0.9	1.2	V
			I_I = 350 µA,	I_C = 200 mA		1	1.4	
			I_I = 500 µA,	I_C = 350 mA		1.2	1.7	
I_{CEX}	Collector cutoff current	Figure 1	V_{CE} = 50 V,	I_I = 0			100	µA
V_F	Clamp forward voltage	Figure 8	I_F = 350 mA			1.7	2.2	V
$I_{I(off)}$	Off-state input current	Figure 3	V_{CE} = 50 V,	I_C = 500 µA	30	65		µA
I_I	Input current	Figure 4	V_I = 3.85 V			0.93	1.35	mA
I_R	Clamp reverse current	Figure 7	V_R = 50 V				100	µA
C_i	Input capacitance		V_I = 0,	f = 1 MHz		15	25	pF

ELECTRICAL CHARACTERISTICS

over recommended operating conditions (unless otherwise noted)

PARAMETER		TEST FIGURE	TEST CONDITIONS		ULQ2003A			ULQ2004A			UNIT
					MIN	TYP	MAX	MIN	TYP	MAX	
$V_{I(on)}$	On-state input voltage	Figure 6	V_{CE} = 2 V	I_C = 125 mA						5	V
				I_C = 200 mA			2.7			6	
				I_C = 250 mA			2.9				
				I_C = 275 mA						7	
				I_C = 300 mA			3				
				I_C = 350 mA						8	
$V_{CE(sat)}$	Collector-emitter saturation voltage	Figure 5	I_I = 250 µA,	I_C = 100 mA		0.9	1.2		0.9	1.1	V
			I_I = 350 µA,	I_C = 200 mA		1	1.4		1	1.3	
			I_I = 500 µA,	I_C = 350 mA		1.2	1.7		1.2	1.6	
I_{CEX}	Collector cutoff current	Figure 1	V_{CE} = 50 V,	I_I = 0			100			50	µA
		Figure 2	V_{CE} = 50 V, T_A = 70°C	I_I = 0						100	
				V_I = 6 V						500	
V_F	Clamp forward voltage	Figure 8	I_F = 350 mA			1.7	2.3		1.7	2	V
$I_{I(off)}$	Off-state input current	Figure 3	V_{CE} = 50 V, T_A = 70°C,	I_C = 500 µA		65		50	65		µA
I_I	Input current	Figure 4	V_I = 3.85 V			0.93	1.35				mA
			V_I = 5 V						0.35	0.5	
			V_I = 12 V						1	1.45	
I_R	Clamp reverse current	Figure 7	V_R = 50 V	T_A = 25°C			100			50	µA
							100			100	
C_i	Input capacitance		V_I = 0,	f = 1 MHz		15	25		15	25	pF

SWITCHING CHARACTERISTICS

T_A = 25°C

PARAMETER		TEST CONDITIONS	ULN2002A, ULN2003A, ULN2004A			UNIT
			MIN	TYP	MAX	
t_{PLH}	Propagation delay time, low- to high-level output	See Figure 9		0.25	1	µs
t_{PHL}	Propagation delay time, high- to low-level output	See Figure 9		0.25	1	µs
V_{OH}	High-level output voltage after switching	V_S = 50 V, I_O = 300 mA, See Figure 10	V_S – 20			mV

SWITCHING CHARACTERISTICS

T_A = 25°C

PARAMETER		TEST CONDITIONS	ULN2003AI			UNIT
			MIN	TYP	MAX	
t_{PLH}	Propagation delay time, low- to high-level output	See Figure 9		0.25	1	µs
t_{PHL}	Propagation delay time, high- to low-level output	See Figure 9		0.25	1	µs
V_{OH}	High-level output voltage after switching	V_S = 50 V, I_O ≈ 300 mA, See Figure 10	V_S – 20			mV

SWITCHING CHARACTERISTICS

T_A = –40°C to 105°C

PARAMETER		TEST CONDITIONS	ULN2003AI			UNIT
			MIN	TYP	MAX	
t_{PLH}	Propagation delay time, low- to high-level output	See Figure 9		1	10	µs
t_{PHL}	Propagation delay time, high- to low-level output	See Figure 9		1	10	µs
V_{OH}	High-level output voltage after switching	V_S = 50 V, I_O ≈ 300 mA, See Figure 10	V_S – 50			mV

SWITCHING CHARACTERISTICS

over recommended operating conditions (unless otherwise noted)

PARAMETER		TEST CONDITIONS	ULQ2003A, ULQ2004A			UNIT
			MIN	TYP	MAX	
t_{PLH}	Propagation delay time, low- to high-level output	See Figure 9		1	10	µs
t_{PHL}	Propagation delay time, high- to low-level output	See Figure 9		1	10	µs
V_{OH}	High-level output voltage after switching	V_S = 50 V, I_O = 300 mA, See Figure 10	V_S – 20			mV

PARAMETER MEASUREMENT INFORMATION

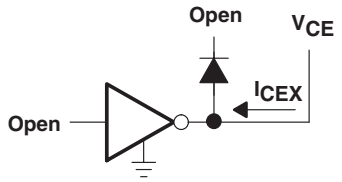


Figure 1. I_{CEX} Test Circuit

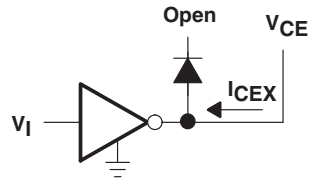


Figure 2. I_{CEX} Test Circuit

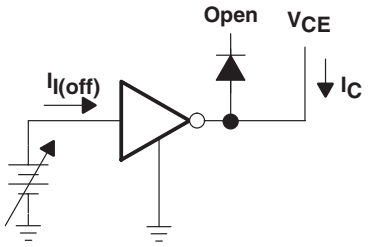


Figure 3. $I_{I(off)}$ Test Circuit

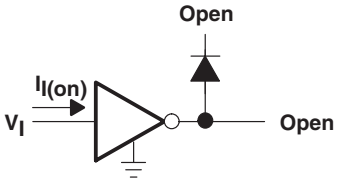


Figure 4. I_I Test Circuit

A. I_I is fixed for measuring $V_{CE(sat)}$, variable for measuring h_{FE}.

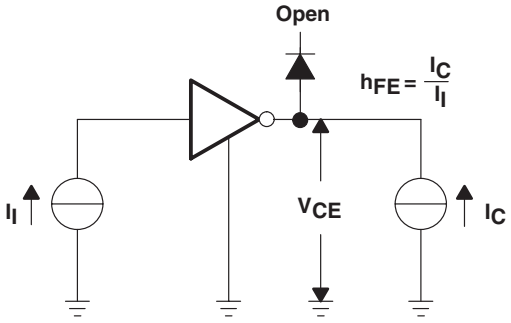


Figure 5. h_{FE}, $V_{CE(sat)}$ Test Circuit

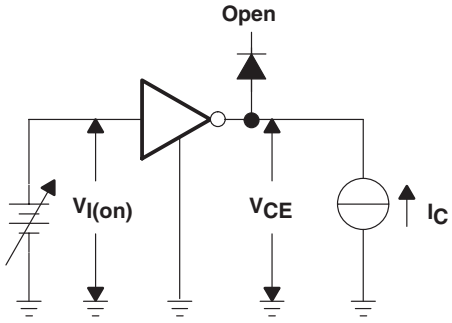


Figure 6. $V_{I(on)}$ Test Circuit

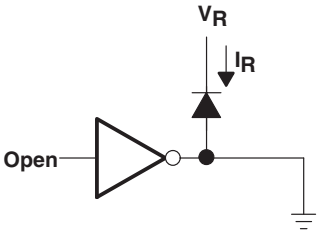


Figure 7. I_R Test Circuit

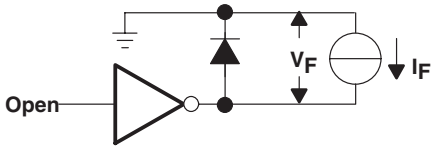


Figure 8. V_F Test Circuit

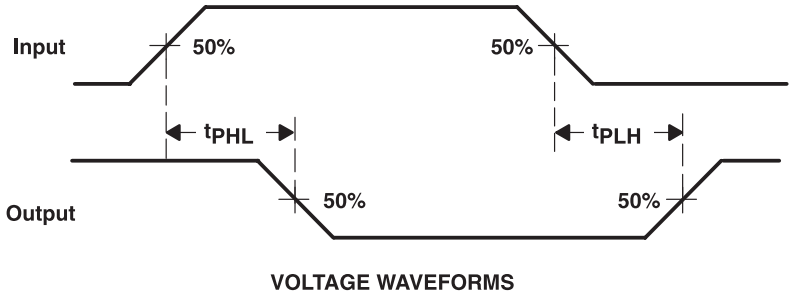
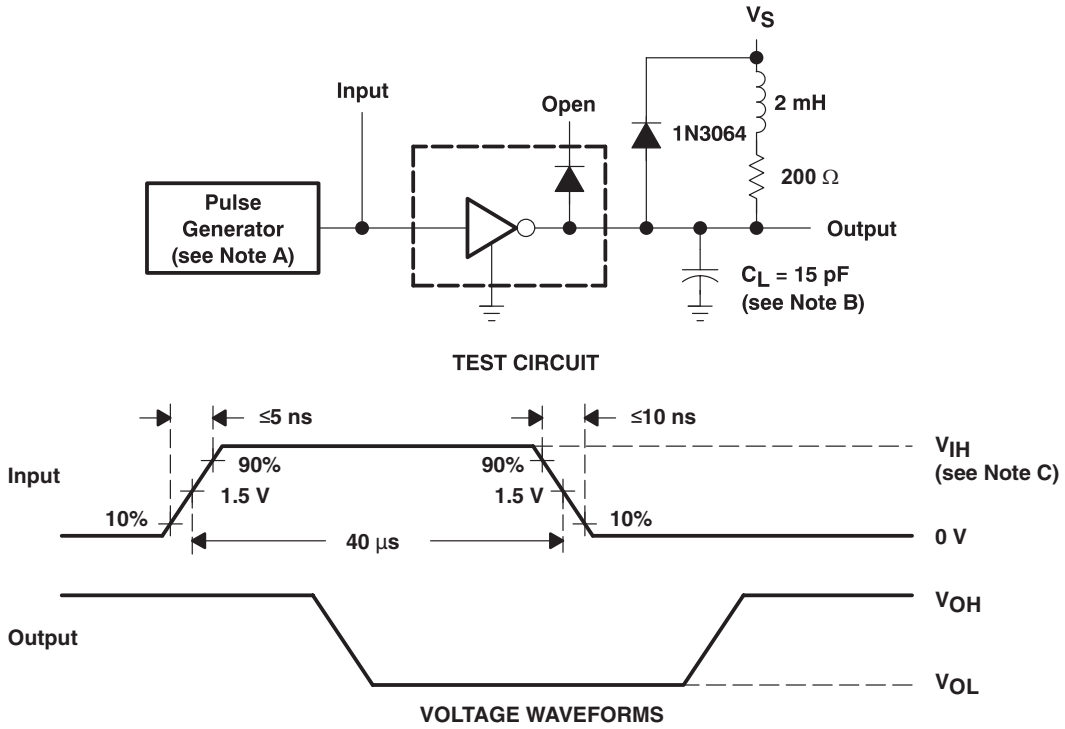


Figure 9. Propagation Delay-Time Waveforms

PARAMETER MEASUREMENT INFORMATION (continued)



A. The pulse generator has the following characteristics: PRR = 12.5 kHz, Z_O = 50 Ω.

B. C_L includes probe and jig capacitance.

C. For testing the ULN2003A, ULN2003AI, and ULQ2003A, V_{IH} = 3 V; for the ULN2002A, V_{IH} = 13 V; for the ULN2004A and the ULQ2004A, V_{IH} = 8 V.

Figure 10. Latch-Up Test Circuit and Voltage Waveforms

TYPICAL CHARACTERISTICS

COLLECTOR-EMITTER SATURATION VOLTAGE vs COLLECTOR CURRENT (ONE DARLINGTON)

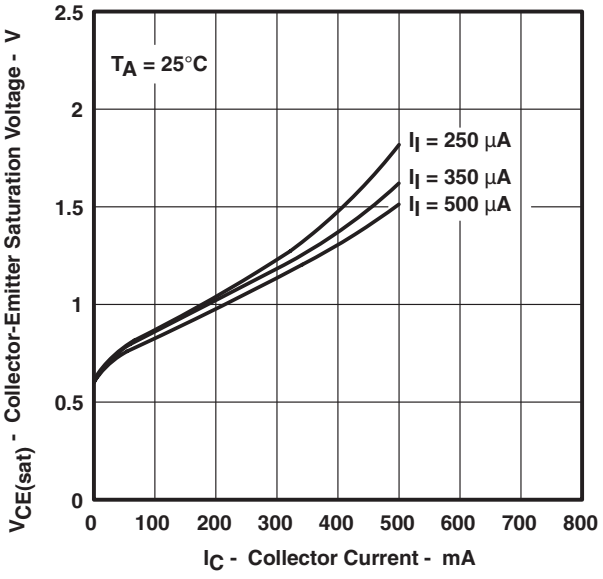


Figure 11.

COLLECTOR-EMITTER SATURATION VOLTAGE vs TOTAL COLLECTOR CURRENT (TWO DARLINGTONS IN PARALLEL)

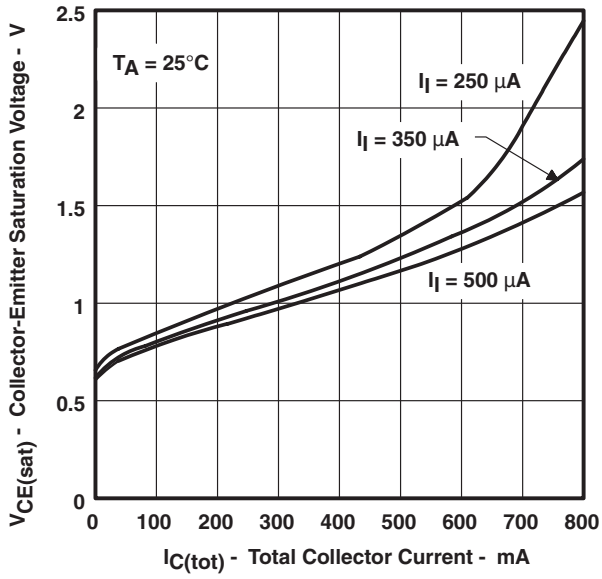


Figure 12.

COLLECTOR CURRENT vs INPUT CURRENT

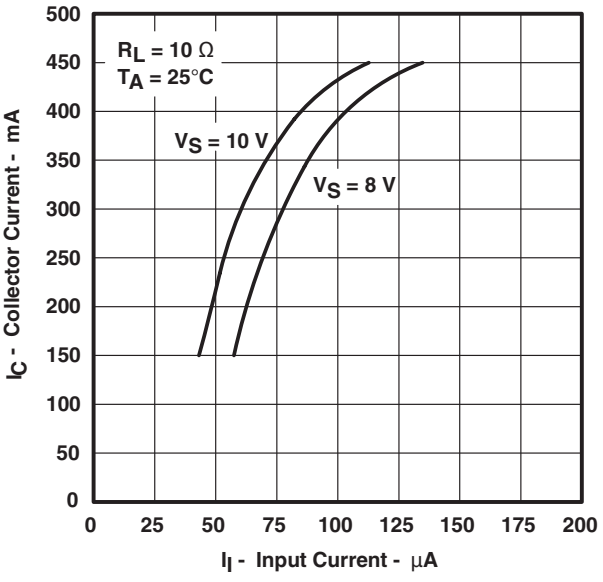


Figure 13.

D PACKAGE MAXIMUM COLLECTOR CURRENT vs DUTY CYCLE

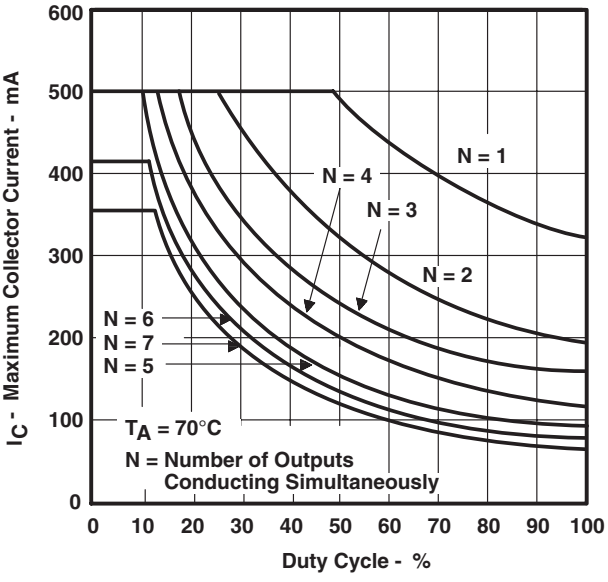


Figure 14.

TYPICAL CHARACTERISTICS (continued)

N PACKAGE
MAXIMUM COLLECTOR CURRENT
vs
DUTY CYCLE

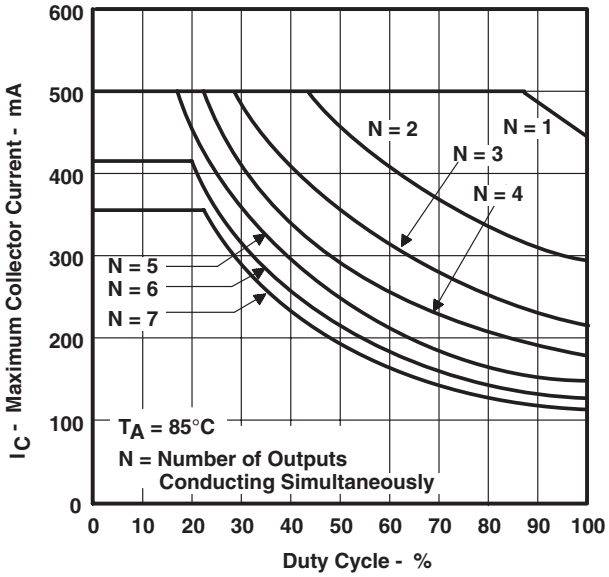


Figure 15.

MAXIMUM AND TYPICAL INPUT CURRENT
vs
INPUT VOLTAGE

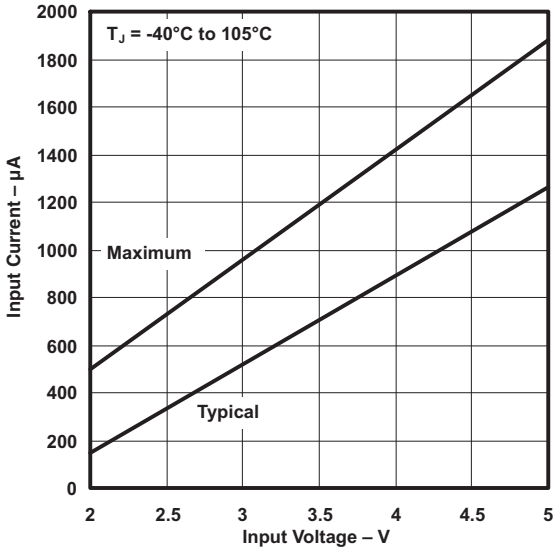


Figure 16.

MAXIMUM AND TYPICAL SATURATED V_{CE}
vs
OUTPUT CURRENT

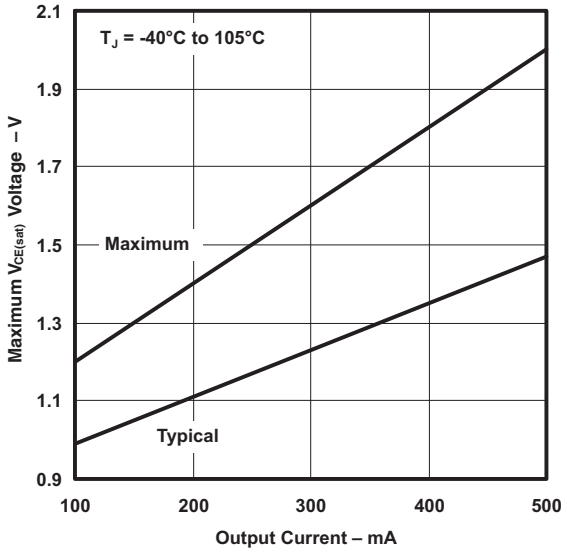


Figure 17.

MINIMUM OUTPUT CURRENT
vs
INPUT CURRENT

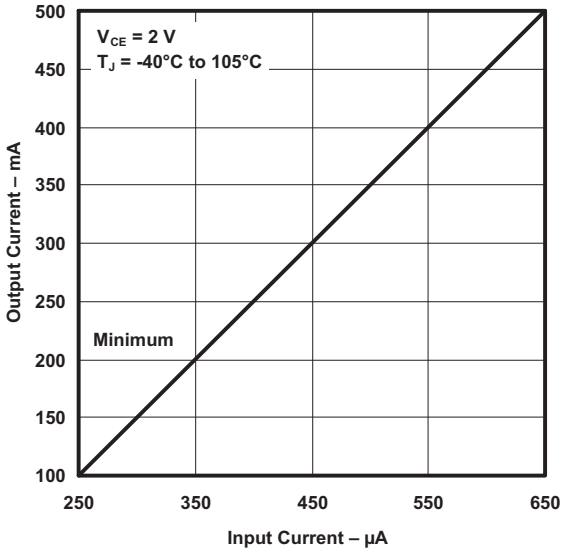


Figure 18.

APPLICATION INFORMATION

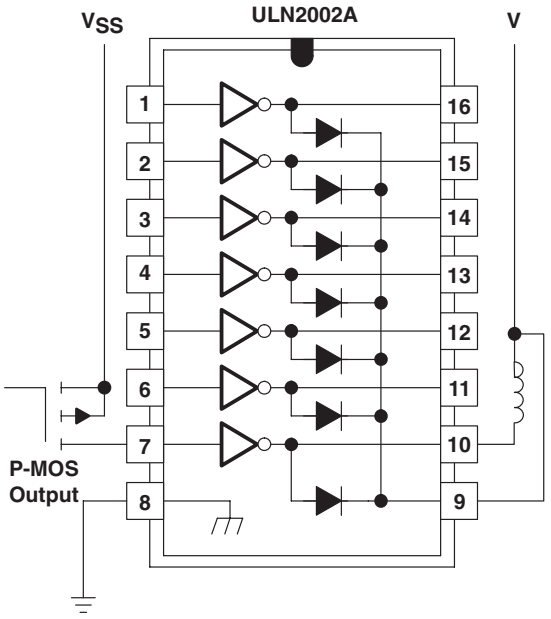


Figure 19. P-MOS to Load

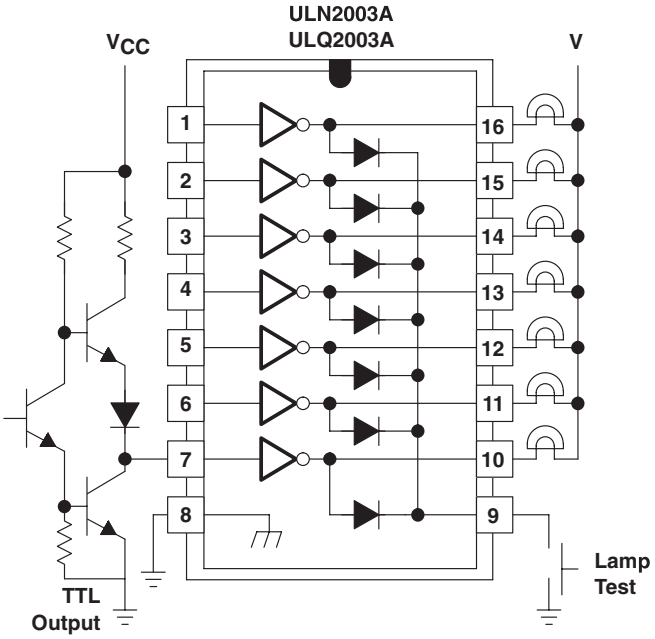


Figure 20. TTL to Load

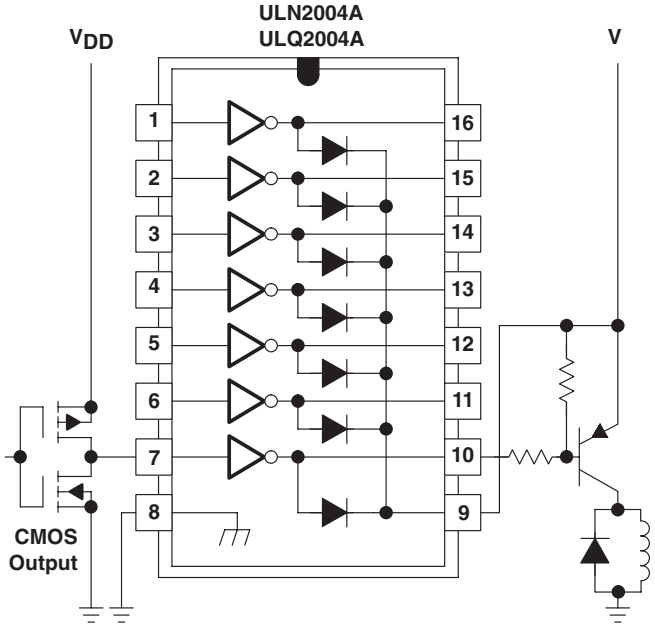


Figure 21. Buffer for Higher Current Loads

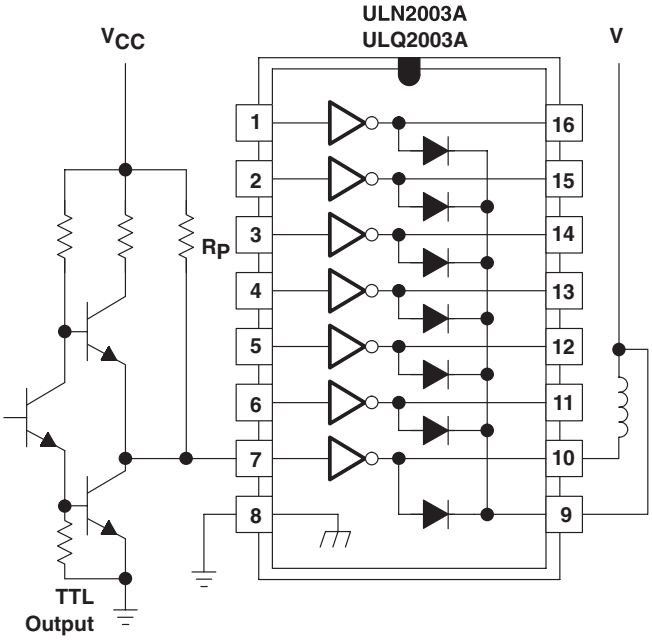


Figure 22. Use of Pullup Resistors to Increase Drive Current

REVISION HISTORY

Changes from Revision J (September 2010) to Revision K **Page**

- Added SOP – NS Package to the –40°C to 85°C T_A range in the Ordering Information table. 2
- Added Input Current (I_I) parameters 5
- Added Input Current (I_I) parameters 6

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/ Ball Finish	MSL Peak Temp (3)	Samples (Requires Login)
ULN2001AD	OBSOLETE	SOIC	D	16		TBD	Call TI	Call TI	
ULN2001ADR	OBSOLETE	SOIC	D	16		TBD	Call TI	Call TI	
ULN2001AN	OBSOLETE	PDIP	N	16		TBD	Call TI	Call TI	
ULN2002AD	OBSOLETE	SOIC	D	16		TBD	Call TI	Call TI	
ULN2002AN	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	
ULN2002ANE4	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	
ULN2003AD	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003ADE4	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003ADG4	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003ADR	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003ADRE4	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003ADRG3	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	
ULN2003ADRG4	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AID	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AIDE4	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AIDG4	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AIDR	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AIDRE4	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AIDRG4	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AIN	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/ Ball Finish	MSL Peak Temp (3)	Samples (Requires Login)
ULN2003AINE4	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	
ULN2003AINSR	ACTIVE	SO	NS	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AIPW	ACTIVE	TSSOP	PW	16	90	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AIPWE4	ACTIVE	TSSOP	PW	16	90	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AIPWG4	ACTIVE	TSSOP	PW	16	90	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AIPWR	ACTIVE	TSSOP	PW	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AIPWRE4	ACTIVE	TSSOP	PW	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AIPWRG4	ACTIVE	TSSOP	PW	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003AJ	OBSOLETE	CDIP	J	16		TBD	Call TI	Call TI	
ULN2003AN	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	
ULN2003ANE3	PREVIEW	PDIP	N	16	25	TBD	Call TI	Call TI	
ULN2003ANE4	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	
ULN2003ANSR	ACTIVE	SO	NS	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003ANSRE4	ACTIVE	SO	NS	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003ANSRG4	ACTIVE	SO	NS	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003APW	ACTIVE	TSSOP	PW	16	90	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003APWE4	ACTIVE	TSSOP	PW	16	90	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003APWG4	ACTIVE	TSSOP	PW	16	90	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003APWR	ACTIVE	TSSOP	PW	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2003APWRE4	ACTIVE	TSSOP	PW	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/ Ball Finish	MSL Peak Temp (3)	Samples (Requires Login)
ULN2003APWRG4	ACTIVE	TSSOP	PW	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2004AD	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2004ADE4	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2004ADG4	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2004ADR	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2004ADRE4	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2004ADRG4	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2004AN	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	
ULN2004ANE4	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	
ULN2004ANSR	ACTIVE	SO	NS	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULN2004ANSRG4	ACTIVE	SO	NS	16	2000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULQ2003AD	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULQ2003ADG4	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULQ2003ADR	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULQ2003ADRG4	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULQ2003AN	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	
ULQ2004AD	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULQ2004ADG4	ACTIVE	SOIC	D	16	40	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULQ2004ADR	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/ Ball Finish	MSL Peak Temp (3)	Samples (Requires Login)
ULQ2004ADRG4	ACTIVE	SOIC	D	16	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	
ULQ2004AN	ACTIVE	PDIP	N	16	25	Pb-Free (RoHS)	CU NIPDAU	N / A for Pkg Type	

(1) The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

(2) Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Pb-Free (RoHS Exempt): This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

(3) MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

OTHER QUALIFIED VERSIONS OF ULQ2003A, ULQ2004A :

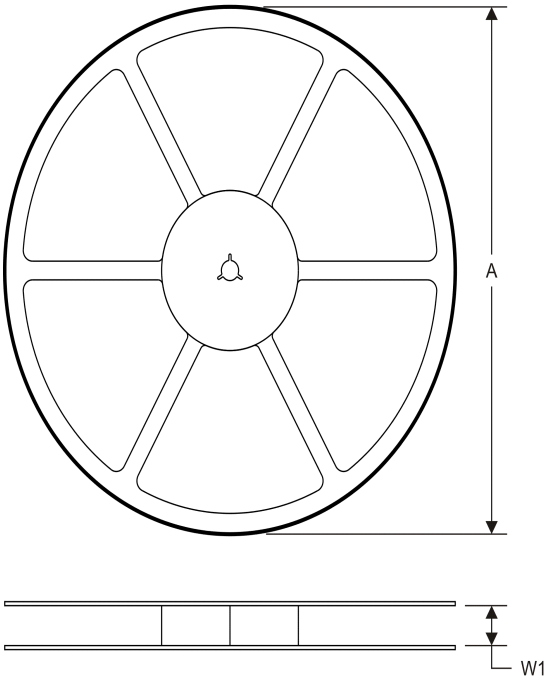
- Automotive: ULQ2003A-Q1, ULQ2004A-Q1

NOTE: Qualified Version Definitions:

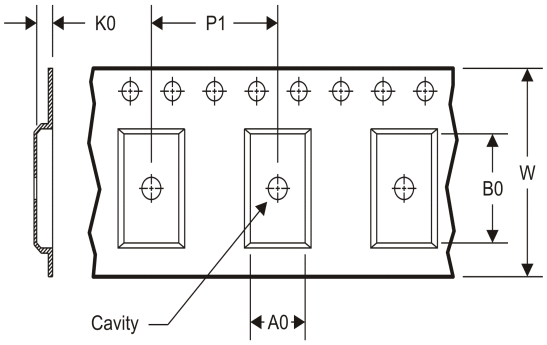
- Automotive - Q100 devices qualified for high-reliability automotive applications targeting zero defects

TAPE AND REEL INFORMATION

REEL DIMENSIONS



TAPE DIMENSIONS



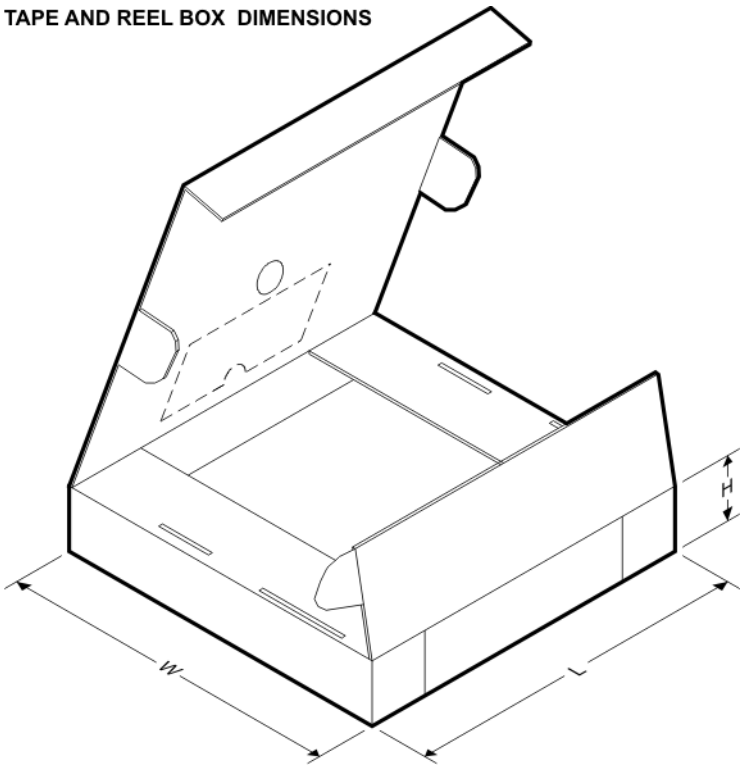
A0	Dimension designed to accommodate the component width
B0	Dimension designed to accommodate the component length
K0	Dimension designed to accommodate the component thickness
W	Overall width of the carrier tape
P1	Pitch between successive cavity centers

TAPE AND REEL INFORMATION

*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Reel Diameter (mm)	Reel Width W1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	P1 (mm)	W (mm)	Pin1 Quadrant
ULN2003ADR	SOIC	D	16	2500	330.0	16.4	6.5	10.3	2.1	8.0	16.0	Q1
ULN2003AIDR	SOIC	D	16	2500	330.0	16.4	6.5	10.3	2.1	8.0	16.0	Q1
ULN2003AINSR	SO	NS	16	2000	330.0	16.4	8.2	10.5	2.5	12.0	16.0	Q1
ULN2003AIPWR	TSSOP	PW	16	2000	330.0	12.4	7.0	5.6	1.6	8.0	12.0	Q1
ULN2003AIPWR	TSSOP	PW	16	2000	330.0	12.4	6.9	5.6	1.6	8.0	12.0	Q1
ULN2003ANSR	SO	NS	16	2000	330.0	16.4	8.2	10.5	2.5	12.0	16.0	Q1
ULN2003APWR	TSSOP	PW	16	2000	330.0	12.4	6.9	5.6	1.6	8.0	12.0	Q1
ULN2004ADR	SOIC	D	16	2500	330.0	16.4	6.5	10.3	2.1	8.0	16.0	Q1
ULN2004ANSR	SO	NS	16	2000	330.0	16.4	8.2	10.5	2.5	12.0	16.0	Q1
ULQ2003ADR	SOIC	D	16	2500	330.0	16.4	6.5	10.3	2.1	8.0	16.0	Q1

TAPE AND REEL BOX DIMENSIONS



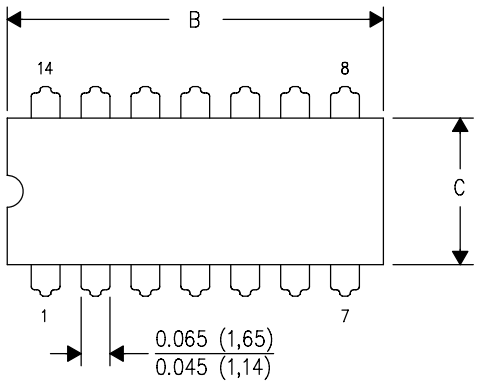
*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Length (mm)	Width (mm)	Height (mm)
ULN2003ADR	SOIC	D	16	2500	333.2	345.9	28.6
ULN2003AIDR	SOIC	D	16	2500	333.2	345.9	28.6
ULN2003AINSR	SO	NS	16	2000	346.0	346.0	33.0
ULN2003AIPWR	TSSOP	PW	16	2000	364.0	364.0	27.0
ULN2003AIPWR	TSSOP	PW	16	2000	346.0	346.0	29.0
ULN2003ANSR	SO	NS	16	2000	346.0	346.0	33.0
ULN2003APWR	TSSOP	PW	16	2000	346.0	346.0	29.0
ULN2004ADR	SOIC	D	16	2500	346.0	346.0	33.0
ULN2004ANSR	SO	NS	16	2000	346.0	346.0	33.0
ULQ2003ADR	SOIC	D	16	2500	333.2	345.9	28.6

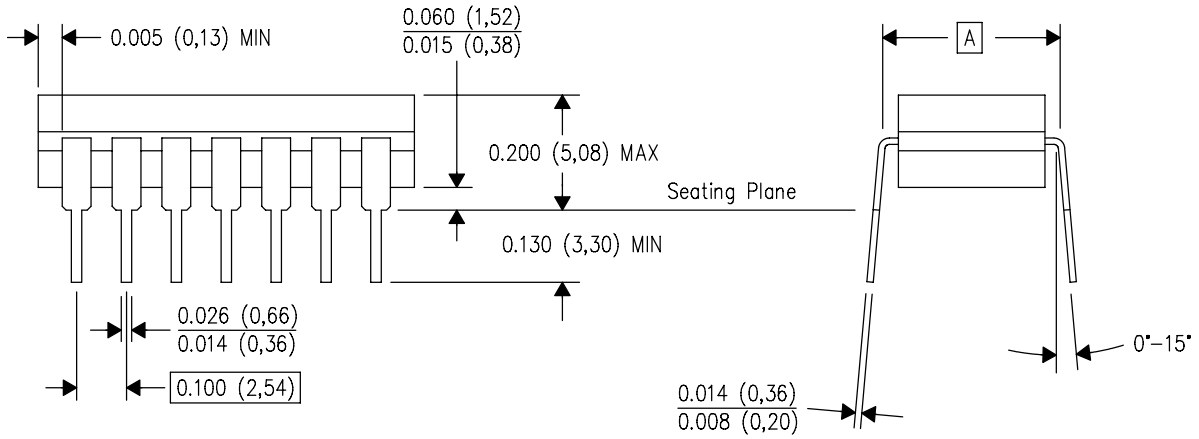
J (R–GDIP–T**)

14 LEADS SHOWN

CERAMIC DUAL IN–LINE PACKAGE



DIM \ PINS **	14	16	18	20
A	0.300 (7,62) BSC	0.300 (7,62) BSC	0.300 (7,62) BSC	0.300 (7,62) BSC
B MAX	0.785 (19,94)	.840 (21,34)	0.960 (24,38)	1.060 (26,92)
B MIN	—	—	—	—
C MAX	0.300 (7,62)	0.300 (7,62)	0.310 (7,87)	0.300 (7,62)
C MIN	0.245 (6,22)	0.245 (6,22)	0.220 (5,59)	0.245 (6,22)



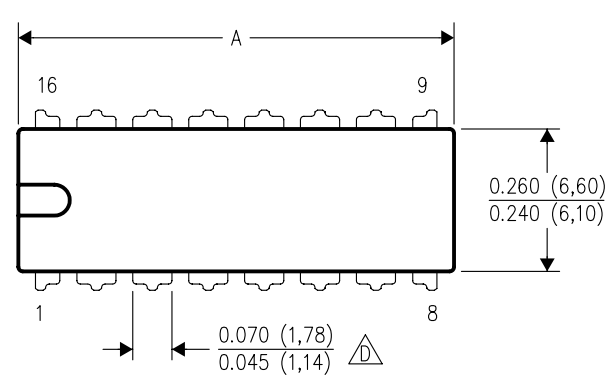
4040083/F 03/03

NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. This package is hermetically sealed with a ceramic lid using glass frit.
D. Index point is provided on cap for terminal identification only on press ceramic glass frit seal only.
E. Falls within MIL STD 1835 GDIP1–T14, GDIP1–T16, GDIP1–T18 and GDIP1–T20.

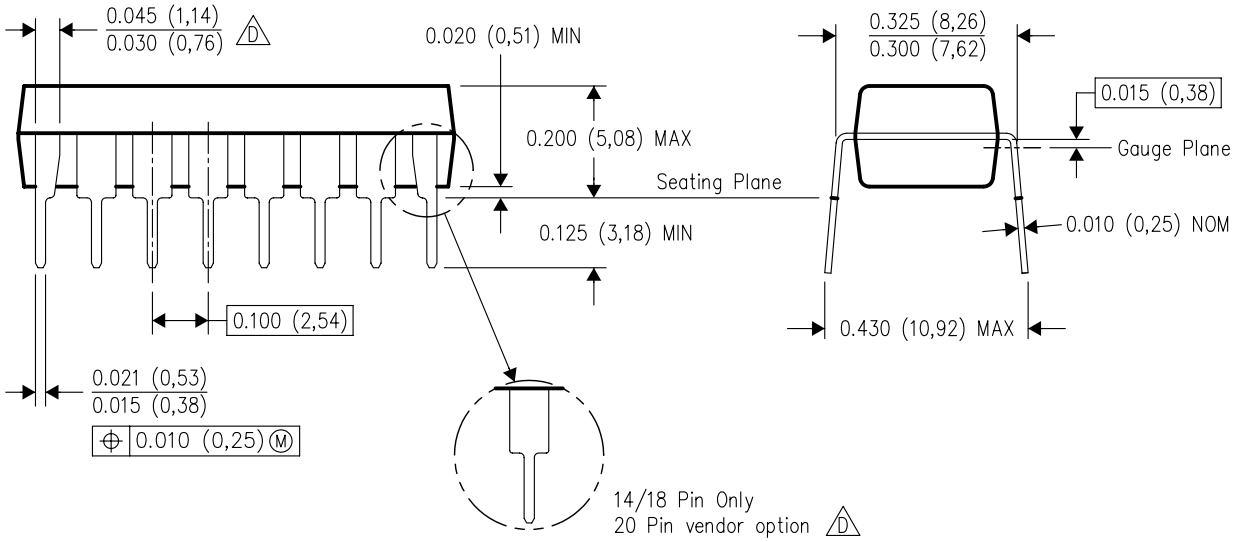
N (R–PDIP–T**)

PLASTIC DUAL–IN–LINE PACKAGE

16 PINS SHOWN



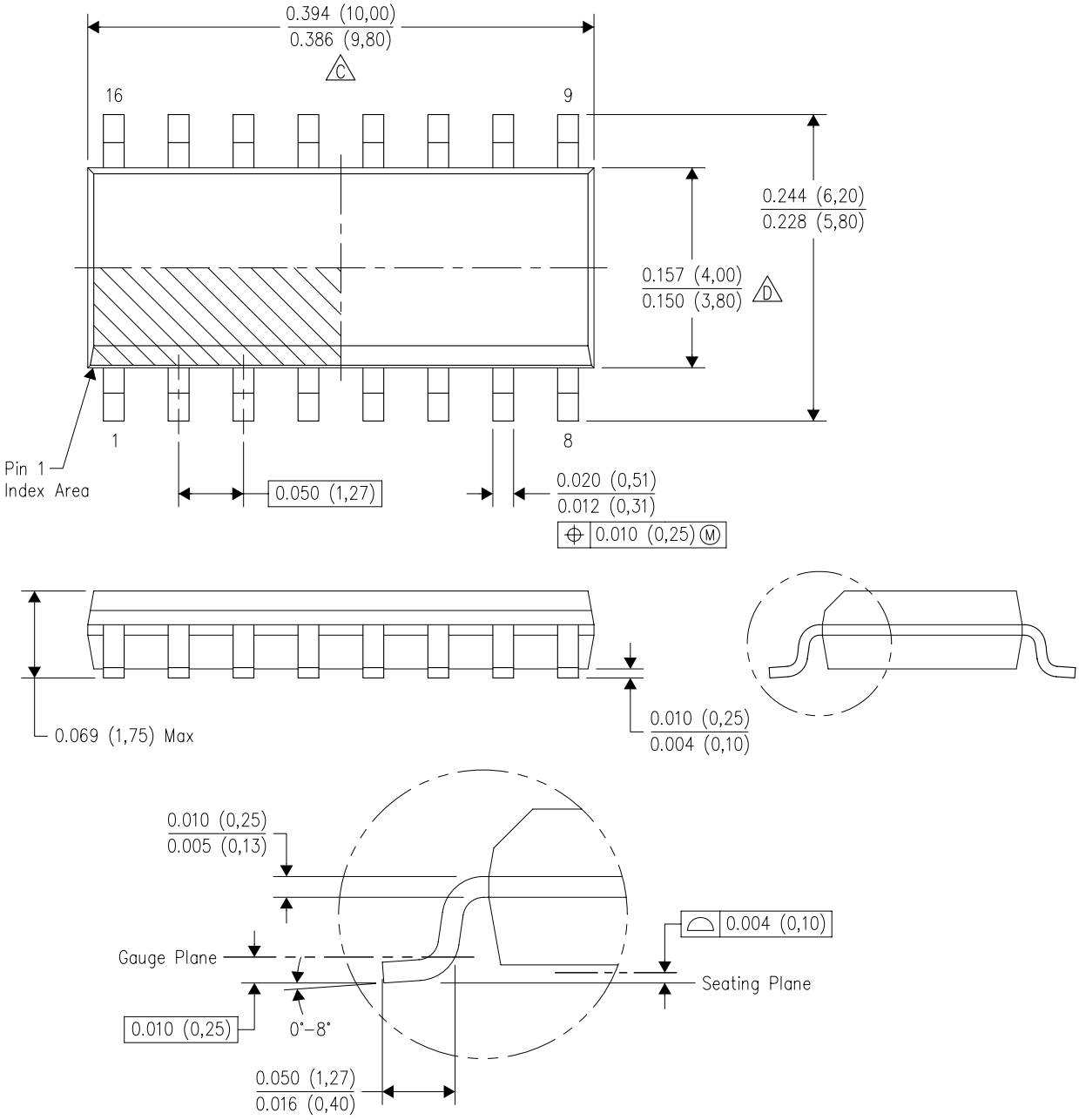
DIM \ PINS **	14	16	18	20
A MAX	0.775 (19,69)	0.775 (19,69)	0.920 (23,37)	1.060 (26,92)
A MIN	0.745 (18,92)	0.745 (18,92)	0.850 (21,59)	0.940 (23,88)
MS–001 VARIATION (C)	AA	BB	AC	AD



4040049/E 12/2002

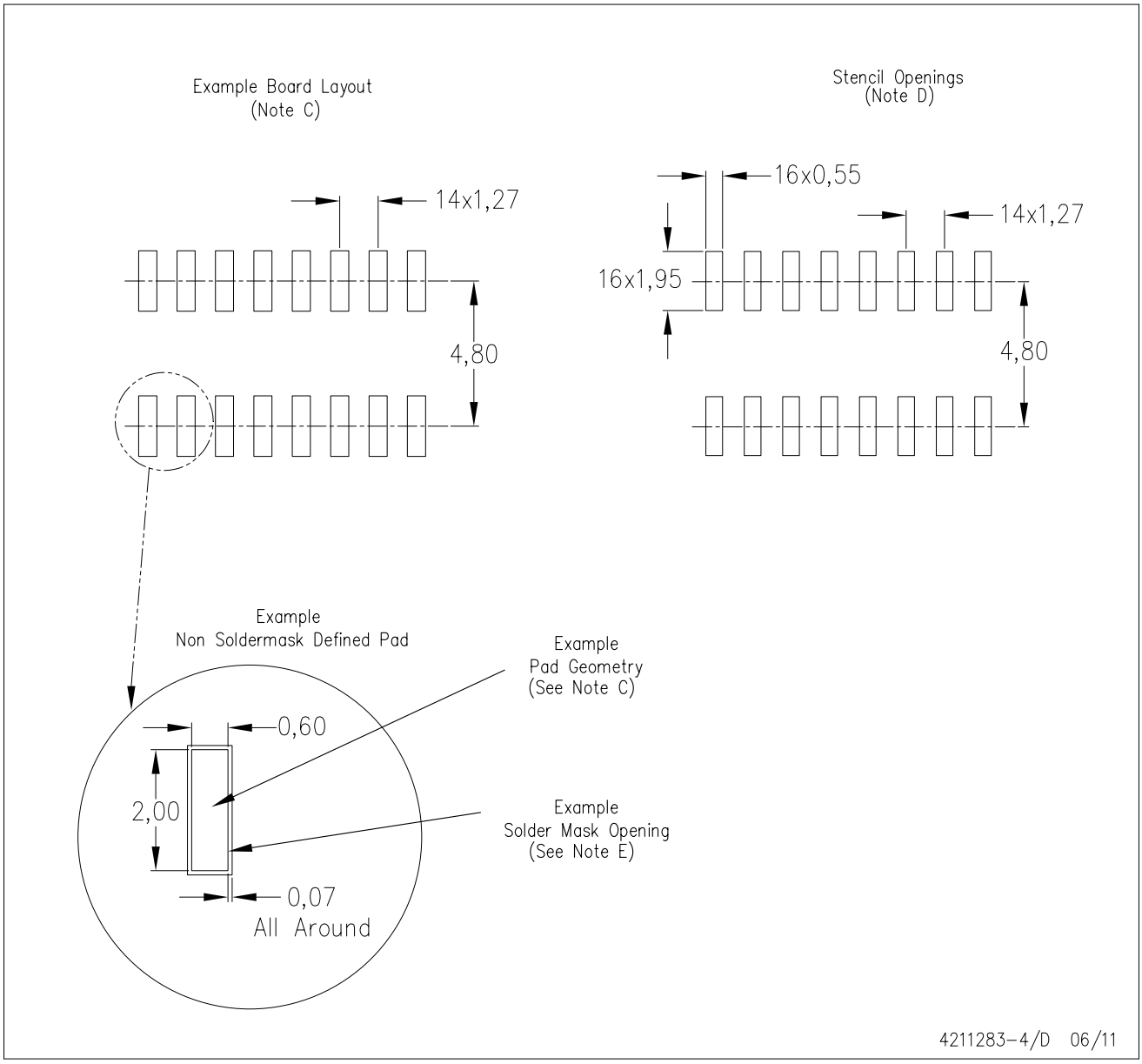
NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Falls within JEDEC MS–001, except 18 and 20 pin minimum body length (Dim A).
D. The 20 pin end lead shoulder width is a vendor option, either half or full width.

D (R–PDSO–G16) PLASTIC SMALL OUTLINE



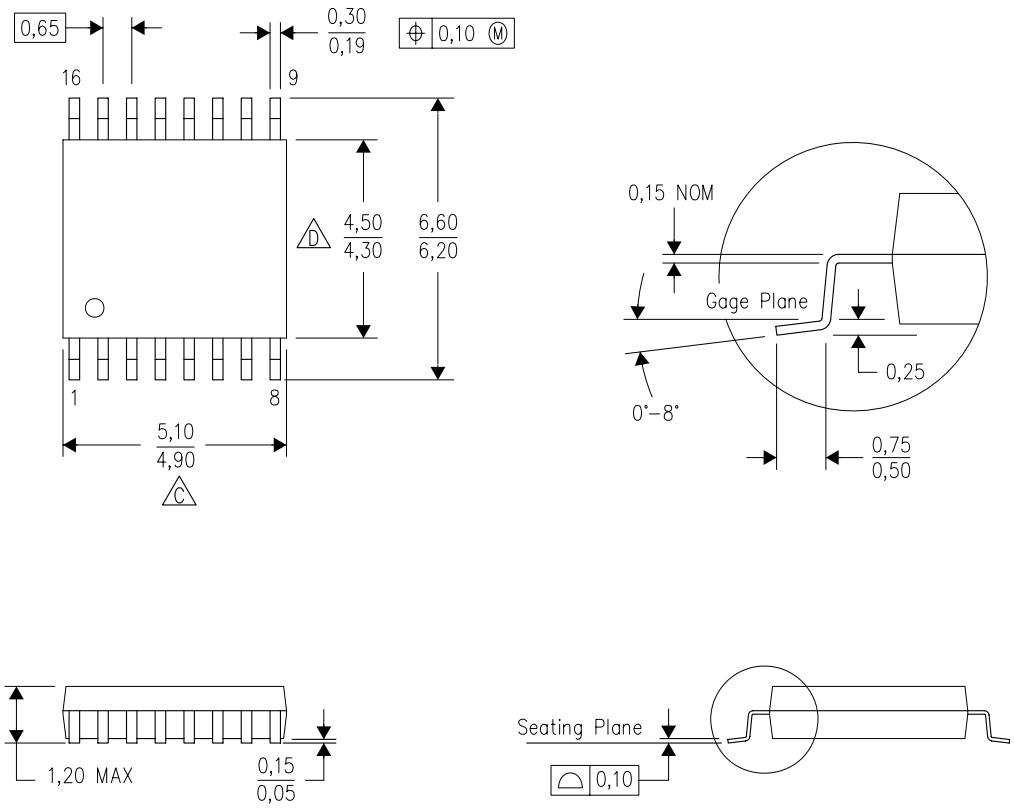
NOTES:
A. All linear dimensions are in inches (millimeters).
B. This drawing is subject to change without notice.
C. Body length does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed 0.006 (0,15) each side.
D. Body width does not include interlead flash. Interlead flash shall not exceed 0.017 (0,43) each side.
E. Reference JEDEC MS–012 variation AC.

D (R–PDSO–G16) PLASTIC SMALL OUTLINE



NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Publication IPC–7351 is recommended for alternate designs.
D. Laser cutting apertures with trapezoidal walls and also rounding corners will offer better paste release. Customers should contact their board assembly site for stencil design recommendations. Refer to IPC–7525 for other stencil recommendations.
E. Customers should contact their board fabrication site for solder mask tolerances between and around signal pads.

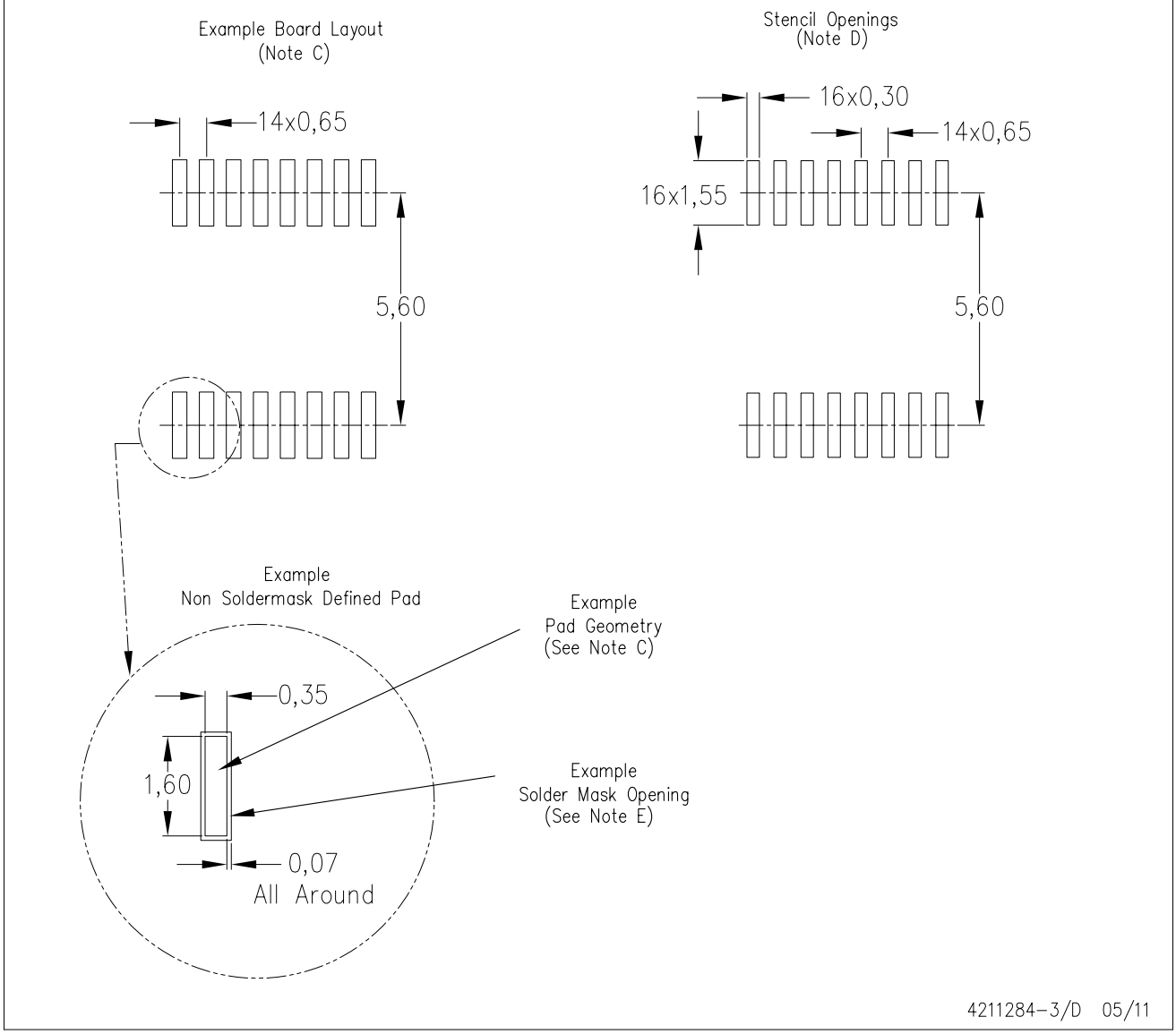
PW (R–PDSO–G16) PLASTIC SMALL OUTLINE



4040064–4/G 02/11

NOTES:
A. All linear dimensions are in millimeters. Dimensioning and tolerancing per ASME Y14.5M–1994.
B. This drawing is subject to change without notice.
C. Body length does not include mold flash, protrusions, or gate burrs. Mold flash, protrusions, or gate burrs shall not exceed 0,15 each side.
D. Body width does not include interlead flash. Interlead flash shall not exceed 0,25 each side.
E. Falls within JEDEC MO–153

PW (R–PDSO–G16) PLASTIC SMALL OUTLINE

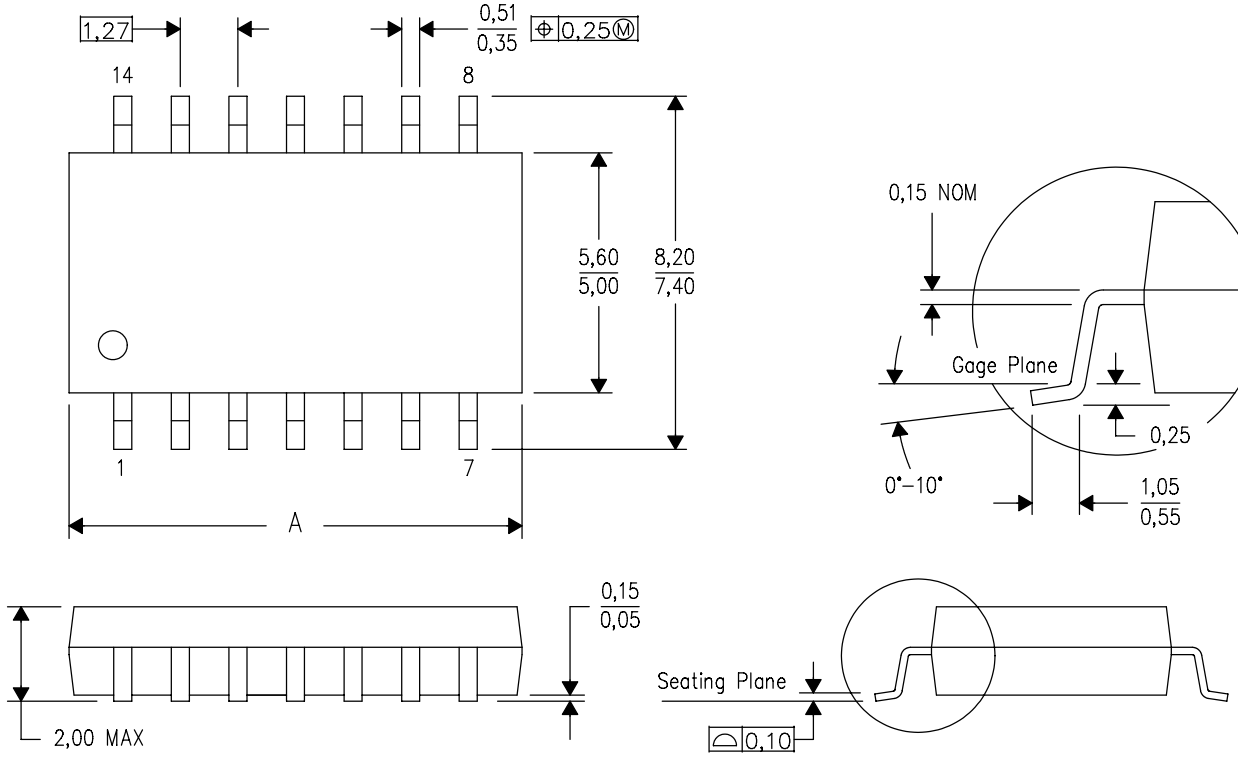


NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Publication IPC–7351 is recommended for alternate designs.
D. Laser cutting apertures with trapezoidal walls and also rounding corners will offer better paste release. Customers should contact their board assembly site for stencil design recommendations. Refer to IPC–7525 for other stencil recommendations.
E. Customers should contact their board fabrication site for solder mask tolerances between and around signal pads.

MECHANICAL DATA

NS (R-PDSO-G)** **PLASTIC SMALL-OUTLINE PACKAGE**

14-PINS SHOWN



DIM \ PINS **	14	16	20	24
A MAX	10,50	10,50	12,90	15,30
A MIN	9,90	9,90	12,30	14,70

4040062/C 03/03

NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion, not to exceed 0,15.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of TI information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation. Information of third parties may be subject to additional restrictions.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Audio	www.ti.com/audio	Communications and Telecom	www.ti.com/communications
Amplifiers	amplifier.ti.com	Computers and Peripherals	www.ti.com/computers
Data Converters	dataconverter.ti.com	Consumer Electronics	www.ti.com/consumer-apps
DLP® Products	www.dlp.com	Energy and Lighting	www.ti.com/energy
DSP	dsp.ti.com	Industrial	www.ti.com/industrial
Clocks and Timers	www.ti.com/clocks	Medical	www.ti.com/medical
Interface	interface.ti.com	Security	www.ti.com/security
Logic	logic.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Power Mgmt	power.ti.com	Transportation and Automotive	www.ti.com/automotive
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Mobile Processors	www.ti.com/omap		
Wireless Connectivity	www.ti.com/wirelessconnectivity		
	TI E2E Community Home Page		e2e.ti.com

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

Datasheet

I2C 1602 Serial LCD Module

Product features:

The I2C 1602 LCD module is a 2 line by 16 character display interfaced to an I2C daughter board. The I2C interface only requires 2 data connections, +5 VDC and GND to operate

For in depth information on I2C interface and history, visit: http://www.wikipedia/wiki/i2c

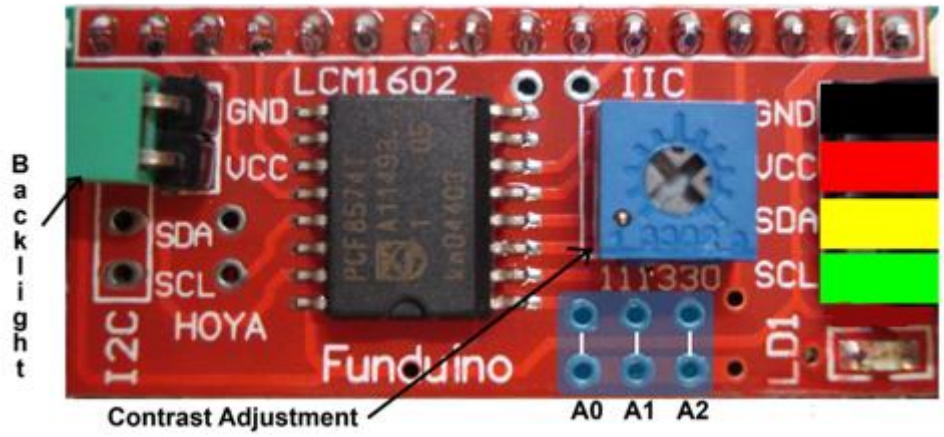
Specifications:

	2 lines by 16 character
I2C Address Range	0x20 to 0x27 (Default=0x27, addressable)
Operating Voltage	5 Vdc
Backlight	White
Contrast	Adjustable by potentiometer on I2c interface
Size	80mm x 36mm x 20 mm
Viewable area	66mm x 16mm

Power:

The device is powered by a single 5Vdc connection.

Pinout Diagram:



Pin/Control Descriptions:

Pin #	Name	Type	Description
1	GND	Power	Supply & Logic ground
2	VCC	Power	Digital I/O 0 or RX (serial receive)
3	SDA	I/O	Serial Data line
4	SCL	CLK	Serial Clock line
A0	A0	Jumper	Optional address selection A0 - see below
A1	A1	Jumper	Optional address selection A1 - see below
A2	A2	Jumper	Optional address selection A2 - see below
Backlight		Jumper	Jumpered - enable backlight, Open - disable backlight
Contrast		Pot	Adjust for best viewing

Addressing:

A0	A1	A2	Address
Open	Open	Open	0x27
Jumper	Open	Open	0x26
Open	Jumper	Open	0x25
Jumper	Jumper	Open	0x24
Open	Open	Jumper	0x23
Jumper	Open	Jumper	0x22
Open	Jumper	Jumper	0x21
Jumper	Jumper	Jumper	0x20

Software:

Download the required LCD Arduino™ library for this device from:

http://www.circuitattic.com/downloads/category/3-sample-code.html?download=9%3Aanother-i2c-library-easier-to-use

Replace current liquid crystal library found in the Arduino library directory with the above
(Note: If you use the examples included with the library, be sure to change address to 0x27)

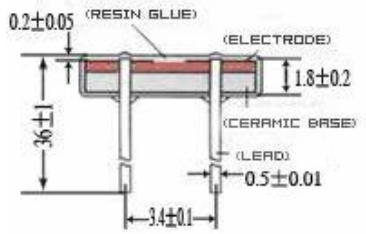
Simple example using library above.

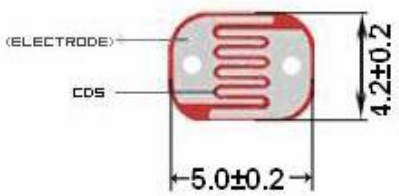
```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#if defined(ARDUINO) && ARDUINO >= 100
#define printByte(args) write(args);
#else
#define printByte(args) print(args,BYTE);
#endif
LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD address to 0x27 for a
//chars and 2 line display
void setup()
{
      lcd.init(); // initialize the lcd
      lcd.backlight();
      lcd.clear();
      delay(100);
      for(int i = 0; i< 3; i++)
      {
            lcd.backlight();
            delay(250);
            lcd.noBacklight();
            delay(250);
      }
      lcd.backlight();


   }
   void loop()
   {
         int x=0;
         lcd.clear();
         lcd.setCursor(2,0); //Start at character 0 on line 0
         lcd.print("Hello World");
         lcd.setCursor(0,1); //Start at character 0 on line 1
         lcd.print(" opencircuit.nl");
         delay(3000); //Wait 3 seconds
         lcd.clear();
         lcd.setCursor(0,0); //Start at character 0 on line 0
         lcd.print("Cursor Blink");
         lcd.blink();
         delay(2000);
         lcd.setCursor(0,0);
         lcd.print("Cursor noBlink");
         lcd.noBlink();
         delay(2000);
   }
```

GM55 Series Datasheet

Photoconductive resistance is a kind of semiconductor resistor, conductivity with light changes. Using the characteristics of different shapes and made by light area of photoconductive resistance. Photoconductive resistance is widely applied in toys, lamps and lanterns, camera, etc.

Structure diagram(unit:mm)





Properties and characteristics

Epoxy encapsulated
Small size
Reliable performance
Quick response
High sensitivity
Good characteristic of spectrum

Type and specification

Specification	Type	Max Voltage (VDC)	Power Dissipation (mw)	Ambient Temperature Range (℃)	Spectral Response peak(nm)	Light Resistance (10Lux) (KΩ)	Dark Resistance MΩ	γ_{10}^{100}	Response time (ms)		Illuminance resistance Characterist
									Incease	Decrease	
Φ 5 Series	GM5516	150	90	-30~+70	540	5-10	0.5	0.5	30	30	1
	GM5528	150	100	-30~+70	540	10-20	1	0.6	20	30	2
	GM5537-1	150	100	-30~+70	540	20-30	2	0.6	20	30	3
	GM5537-2	150	100	-30~+70	540	30-50	3	0.7	20	30	3
	GM5539	150	100	-30~+70	540	50-100	5	0.8	20	30	4
	GM5549	150	100	-30~+70	540	100-200	10	0.9	20	30	5

Measuring Conditions

1. Light Resistance:
   measured at 10 lux with standard light A (2854k color temperature) and 2h pre-illumination at 400-600 lux prior to testing.

2. Dark Resistance:
   measured 10 seconds after pulsed 10 lux.

3. Gamma Characteristic:
   between 10 lux and 100 lux and given by

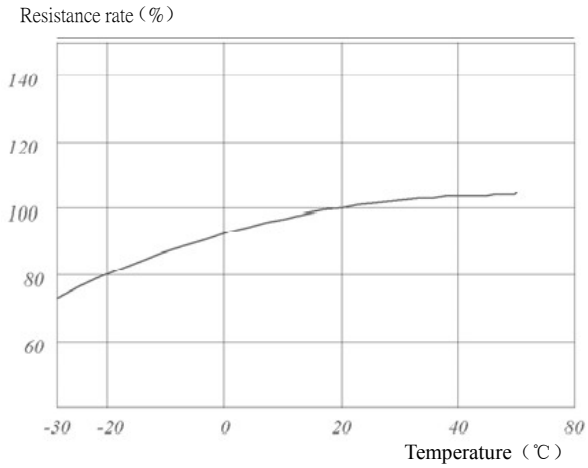
   $$T = \frac{\log (R10/R100)}{\log (100/10)} = \log (R10/R100)$$

   R10, R100 cell resistance at 10 lux and 100 lux.
   The error of T is +0.1.

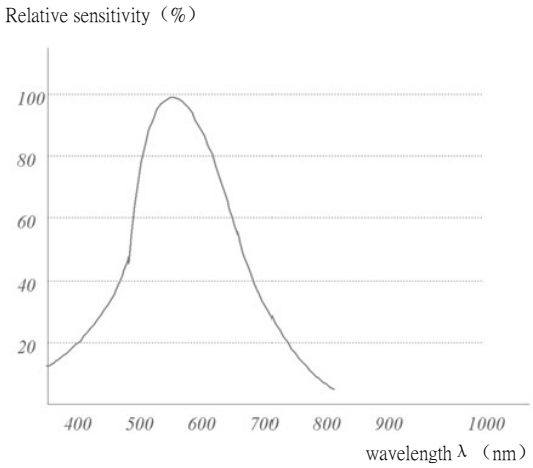
4. Pmax:
   Max. power dissipation at ambient temperature of 25°C.

5. Vmax:
   Max. voltage in darkness that may be applied to the cell continuously.

Main characteristic curve



Temperature characteristic curve



Spectral response characteristic curve

Light - resistance characteristic curve

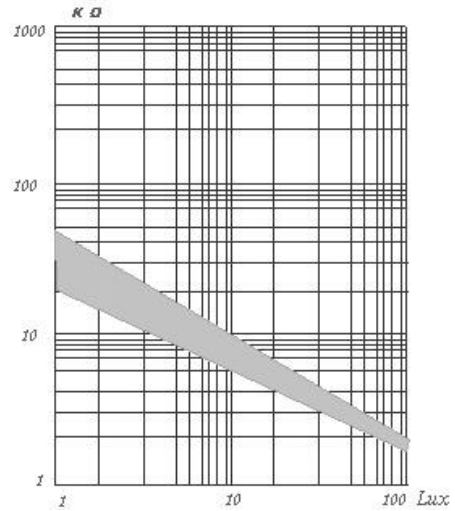


Figure 1

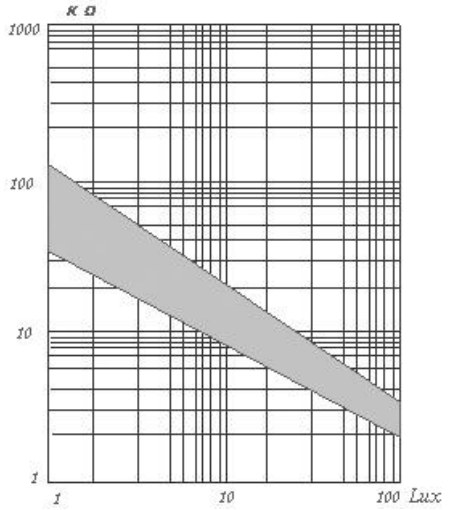


Figure 2

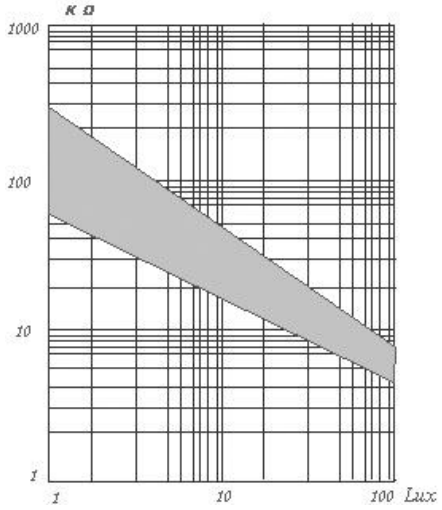


Figure 3

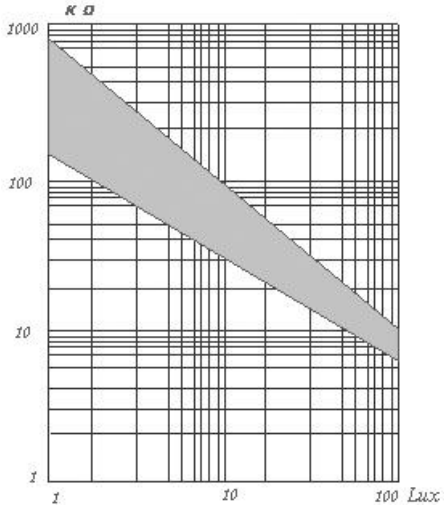


Figure 4

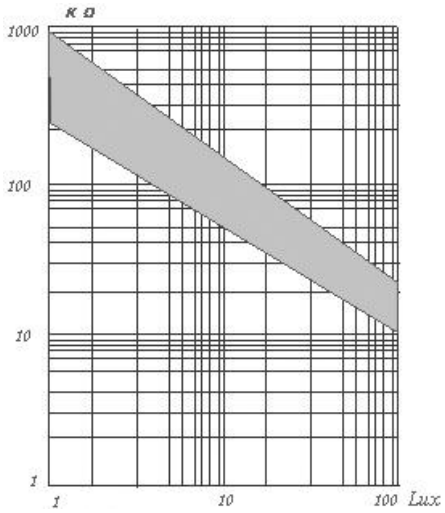


Figure 5

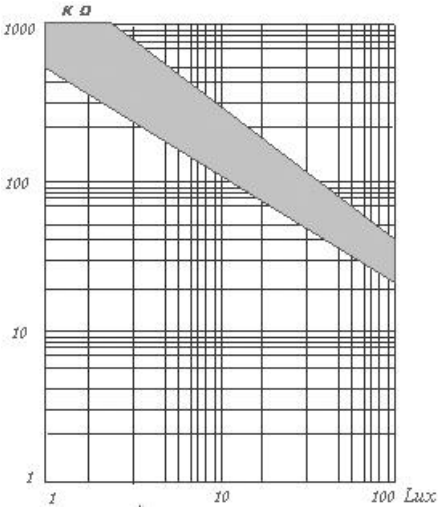


Figure 6

Note

This product adopts environmental materials packaging,little packaging is 200pcs, big packaging is 2000pcs.

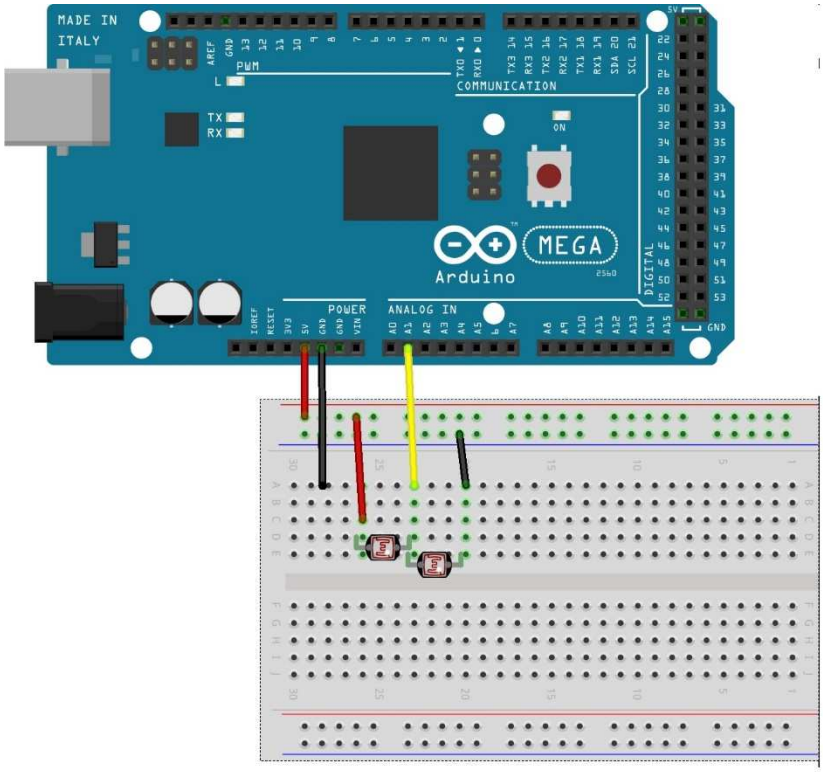
To avoid this product in damp,high temperature environment preservation.

Welding possible time.

Attention should be apart from ceramic welding wire 4mm base location.

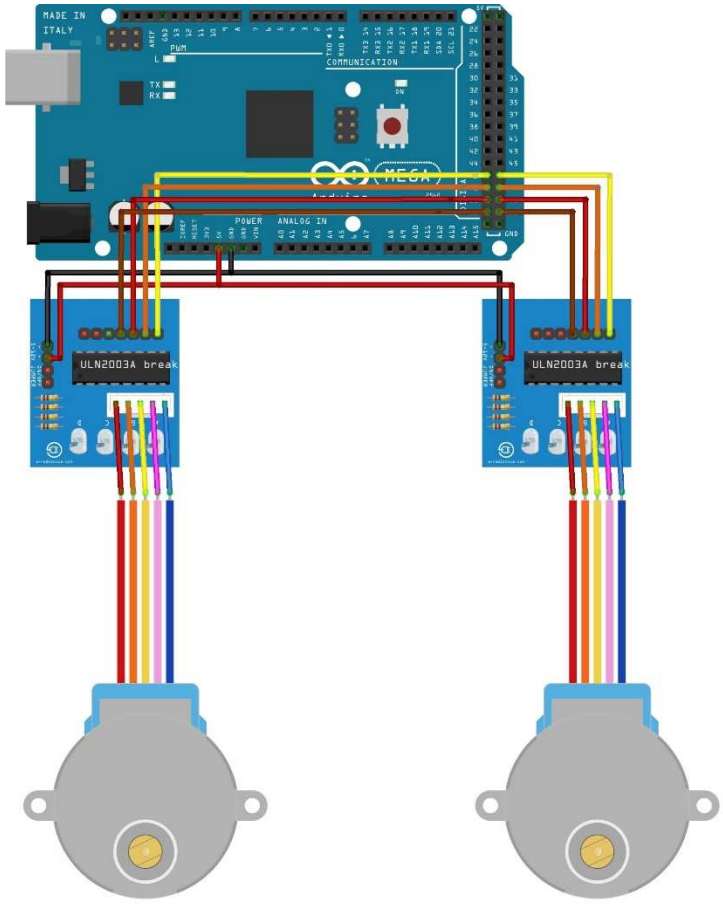
A2. Esquemes de connexionat

A2.1 Proves LDR



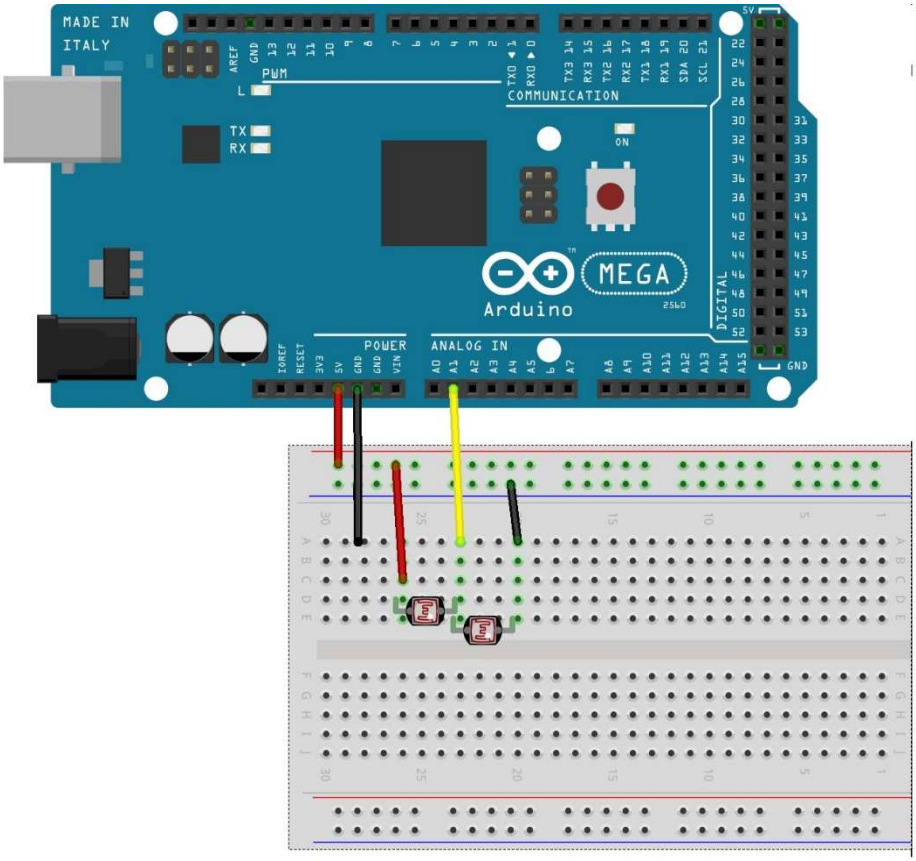
Imatge A2.1. Esquema proves LDR

A2.2 Proves PAP



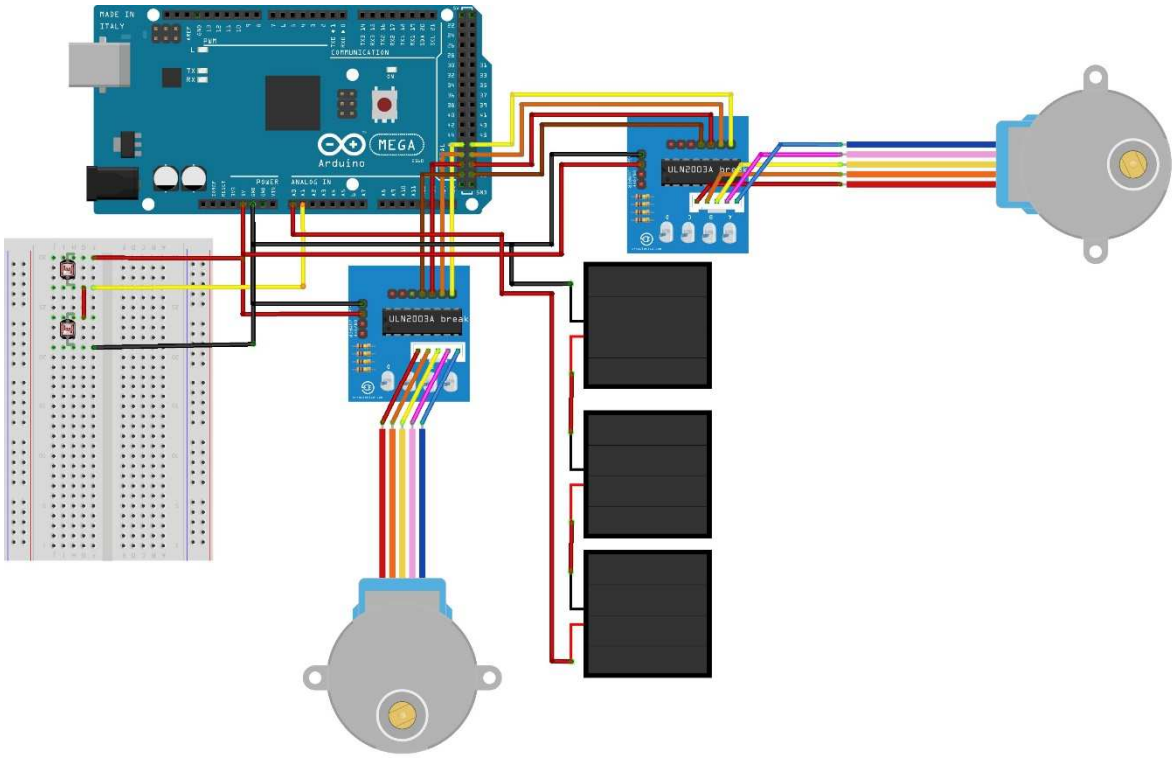
Imatge A2.2. Esquema proves Motors PAP

A2.3 Seguidor LDR



Imatge A2.3. Esquema Seguidor LDR

A2.4 Seguidor PV



Imatge A2.4. Esquema Seguidor PV

Annex B: Manuals d'ús

B1. Manual d'ús d'Arduino

arduino programming notebook

brian w. Evans

edición española

Traducción:
José Manuel Ruiz Gutiérrez
Adaptación:
José Manuel Escuder Martinez

http://www.ardumania.es/

ver. 1.2 de 18/08/2011

Datos del documento original

Arduino Notebook: A Beginner's Reference Written and compiled by Brian W. Evans

With information or inspiration taken from:
http://www.arduino.cc
http://www.wiring.org.co
http://www.arduino.cc/en/Booklet/HomePage (enlace roto)
http://cslibrary.stanford.edu/101/

Including material written by:
Massimo Banzi
Hernando Barragán
David Cuartielles
Tom Igoe
Todd Kurt
David Mellis and others

Published:
First Edition August 2007

contenido

prefacio

El propósito del autor original de este libro fue crear un pequeño manual de consulta rápida sobre los comandos básicos y la sintaxis del lenguaje de programación de Arduino. Para entrar en los contenidos con mayor profundidad se pueden consultar otras páginas web, libros, workshops y cursos. Esta decisión hizo que quedaran fuera del contenido formas complejas como los arrays o avanzadas formas de comunicación serie.

Comenzando con la estructura básica del C del que deriva el lenguaje de programación de Arduino este libro de notas continua con la descripción de los comandos más usuales e ilustra su uso con ejemplos de código.

Esta traducción al español la realizó en su día José Manuel Ruiz Gutierrez para utilizarla dentro de sus cursos. Entre 2010 y 2011 la comunidad de traductores quisimos ampliar la documentación existente en Español para beneficio de todos los usuarios que tienen problemas con el ingles. Lamentablemente problemas derivados de la forma de trabajar de una comunidad formada exclusivamente por voluntarios dejaron este proyecto congelado.

Respetando los términos de la licencia del documento original este libro ha sido remaquetado para adaptarlo al Español. No debe considerarse una obra cerrada, si no que espero publicar futuras revisiones ampliando su contenido y adecuándolo a las novedades acontecidas en este mundillo desde la primera edición de este libro.

Cualquier aportación, corrección o sugerencia puede ser enviada a: josemescuder@gmail.com

estructura de un sketch

La estructura básica del lenguaje de programación de Arduino es bastante simple y se compone de al menos dos partes. Estas dos partes necesarias, o funciones, encierran bloques que contienen declaraciones, estamentos o instrucciones.

```
void setup()
{
   estamentos;
}
void loop()
{
   estamentos;
}
```

En donde setup() es la parte encargada de recoger la configuración y loop() es la que contienen el programa que se ejecutará cíclicamente (de ahí el termino loop –bucle-). Ambas funciones son necesarias para que el programa trabaje.

La función de configuración debe contener la declaración de las variables. Es la primera función a ejecutar en el programa, se ejecuta sólo una vez, y se utiliza para configurar o inicializar pinMode (modo de trabajo de las E/S), configuración de la comunicación en serie y otras.

La función bucle (loop) siguiente contiene el código que se ejecutara continuamente (lectura de entradas, activación de salidas, etc) Esta función es el núcleo de todos los programas de Arduino y la que realiza la mayor parte del trabajo.

setup()

La función setup() se invoca una sola vez cuando el programa empieza. Se utiliza para inicializar los modos de trabajo de los pins, o el puerto serie. Debe ser incluido en un programa aunque no haya declaración que ejecutar. Así mismo se puede utilizar para establecer el estado inicial de las salidas de la placa.

```
void setup()
```

```
{
   pinMode(pin, OUTPUT); // configura el 'pin' como
salida
   digitalWrite(pin, HIGH);  // pone el 'pin' en estado
                             // HIGH
}
```

loop()

Después de llamar a setup(), la función loop() hace precisamente lo que sugiere su nombre, se ejecuta de forma cíclica, lo que posibilita que el programa este respondiendo continuamente ante los eventos que se produzcan en la placa.

```
void loop()
{
digitalWrite(pin, HIGH); // pone en uno (on, 5v) el´pin´
delay(1000);             // espera un segundo (1000 ms)
digitalWrite(pin, LOW); // pone en cero (off, 0v.) el
delay(1000);             // ´pin´
}
```

funciones

Una función es un bloque de código que tiene un nombre y un conjunto de instrucciones que son ejecutadas cuando se llama a la función. Son funciones setup() y loop() de las que ya se ha hablado. Las funciones de usuario pueden ser escritas para realizar tareas repetitivas y para reducir el tamaño de un programa. Las funciones se declaran asociadas a un tipo de valor. Este valor será el que devolverá la función, por ejemplo 'int' se utilizará cuando la función devuelva un dato numérico de tipo entero. Si la función no devuelve ningún valor entonces se colocará delante la palabra “void”, que significa “función vacía”. Después de declarar el tipo de dato que devuelve la función se debe escribir el nombre de la función y entre paréntesis se escribirán, si es necesario, los parámetros que se deben pasar a la función para que se ejecute.

```
tipo nombreFunción(parámetros)
{
instrucciones;
}
```

La función siguiente devuelve un número entero, delayVal() se utiliza para poner un valor de retraso en un programa que lee una variable analógica de un potenciómetro conectado a una entrada de Arduino. Al principio se declara como una variable local, ´v´ recoge el valor leído del potenciómetro que estará comprendido entre 0 y 1023, luego se divide el valor por 4 para ajustarlo a un margen comprendido entre 0 y 255, finalmente se devuelve el valor ´v´ y se retornaría al programa principal.

```
int delayVal()
{
int v;                  // crea una variable temporal 'v'
v= analogRead(pot);     // lee el valor del potenciómetro
v /= 4;                  // convierte 0-1023 a 0-255
return v;                // devuelve el valor final
}
```

{} entre llaves

Las llaves sirven para definir el principio y el final de un bloque de instrucciones. Se utilizan para los bloques de programación setup(), loop(), if.., etc.

```
type funcion()
{
instrucciones;
}
```

Una llave de apertura "{" siempre debe ir seguida de una llave de cierre "}", si no es así el compilador dará errores.

El entorno de programación de Arduino incluye una herramienta de gran utilidad para comprobar el total de llaves. Sólo tienes que hacer click en el punto de inserción de una llave abierta e inmediatamente se marca el correspondiente cierre de ese bloque (llave cerrada).

; punto y coma

El punto y coma ";" se utiliza para separar instrucciones en el lenguaje de programación de Arduino. También se utiliza para separar elementos en una instrucción de tipo "bucle for".

```
int x = 13;   // declara la variable 'x' como tipo
              // entero de valor 13
```

Nota: Si olvidáis poner fin a una línea con un punto y coma se producirá en un error de compilación. El texto de error puede ser obvio, y se referirá a la falta de un punto y coma, o puede que no. Si se produce un error raro y de difícil detección lo primero que debemos hacer es comprobar que los puntos y comas están colocados al final de las instrucciones.

/*… */ bloque de comentarios

Los bloques de comentarios, o comentarios multi-línea son áreas de texto ignorados por el programa que se utilizan para las descripciones del código o comentarios que ayudan a comprender el programa. Comienzan con / * y terminan con * / y pueden abarcar varias líneas.

```
/* esto es un bloque de comentario no se debe olvidar
cerrar los comentarios estos deben estar equilibrados */
```

Debido a que los comentarios son ignorados por el compilador y no ocupan espacio en la memoria de Arduino pueden ser utilizados con generosidad. También pueden utilizarse para "comentar" bloques de código con el propósito de anotar informaciones para depuración y hacerlo mas comprensible para cualquiera.

Nota: Dentro de una misma línea de un bloque de comentarios no se puede escribir otra bloque de comentarios (usando /*..*/).

// línea de comentarios

Una línea de comentario empieza con / / y terminan con la siguiente línea de código. Al igual que los comentarios de bloque, los de línea son ignoradas por el programa y no ocupan espacio en la memoria.

```
// esto es un comentario
```

Una línea de comentario se utiliza a menudo después de una instrucción, para proporcionar más información acerca de lo que hace esta o para recordarla más adelante.

variables

Una variable es una manera de nombrar y almacenar un valor numérico para su uso posterior por el programa. Como su nombre indica, las variables son números que se pueden variar continuamente en contra de lo que ocurre con las constantes cuyo valor nunca cambia. Una variable debe ser declarada y, opcionalmente, asignarle un valor. El siguiente código de ejemplo declara una variable llamada variableEntrada y luego le asigna el valor obtenido en la entrada analógica del PIN2:

```
int variableEntrada = 0;   // declara una variable y le
                           // asigna el valor 0
variableEntrada = analogRead(2);  // la variable recoge
                           //el valor analógico del PIN2
```

'variableEntrada' es la variable en sí. La primera línea declara que será de tipo entero “int”. La segunda línea fija a la variable el valor correspondiente a la entrada analógica PIN2. Esto hace que el valor de PIN2 sea accesible en otras partes del código.
Una vez que una variable ha sido asignada, o re-asignada, usted puede probar su valor para ver si cumple ciertas condiciones, o puede utilizar directamente su valor. Como ejemplo ilustrativo veamos tres operaciones útiles con variables: el siguiente código prueba si la variable “entradaVariable” es inferior a 100, si es cierto se asigna el valor 100 a “entradaVariable” y, a continuación, establece un retardo (delay) utilizando como valor “entradaVariable” que ahora será como mínimo de valor 100:

```
if (entradaVariable < 100) // pregunta si la variable es
{                          //menor de 100
  entradaVariable = 100; // si es cierto asigna el valor
}                          //100
delay(entradaVariable);   // usa el valor como retardo
```

Nota: Las variables deben tomar nombres descriptivos, para hacer el código más legible. Los nombres de variables pueden ser “contactoSensor” o “pulsador”, para ayudar al programador y a cualquier otra persona a leer el código y entender lo que representa la variable. Nombres de variables como “var” o “valor”, facilitan muy poco que el código sea inteligible. Una variable puede ser cualquier

nombre o palabra que no sea una palabra reservada en el entorno de Arduino.

declaración de variables

Todas las variables tienen que declararse antes de que puedan ser utilizadas. Para declarar una variable se comienza por definir su tipo como int (entero), long (largo), float (coma flotante), etc, asignándoles siempre un nombre, y, opcionalmente, un valor inicial. Esto sólo debe hacerse una vez en un programa, pero el valor se puede cambiar en cualquier momento usando aritmética y reasignaciones diversas.

El siguiente ejemplo declara la variable entradaVariable como una variable de tipo entero “int”, y asignándole un valor inicial igual a cero. Esto se llama una asignación.

```
int entradaVariable = 0;
```

Una variable puede ser declarada en una serie de lugares del programa y en función del lugar en donde se lleve a cabo la definición esto determinará en que partes del programa se podrá hacer uso de ella.

utilización de una variable

Una variable puede ser declarada al inicio del programa antes de la parte de configuración setup(), a nivel local dentro de las funciones, y, a veces, dentro de un bloque, como para los bucles del tipo if.. for.., etc. En función del lugar de declaración de la variable así se determinara el ámbito de aplicación, o la capacidad de ciertas partes de un programa para hacer uso de ella.

Una variable global es aquella que puede ser vista y utilizada por cualquier función y estamento de un programa. Esta variable se declara al comienzo del programa, antes de setup().

Una variable local es aquella que se define dentro de una función o como parte de un bucle. Sólo es visible y sólo puede utilizarse dentro de la función en la que se declaró.

Por lo tanto, es posible tener dos o más variables del mismo nombre en diferentes partes del mismo programa que pueden contener valores diferentes. La garantía de que sólo una función tiene acceso a sus variables dentro del programa simplifica y reduce el potencial de errores de programación.

El siguiente ejemplo muestra cómo declarar a unos tipos diferentes de variables y la visibilidad de cada variable:

```
int value;    // 'value' es visible para cualquier
función

void setup()
{
// no es necesario configurar nada en este ejemplo
}
void loop()
{
   for (int i=0; i<20;)    // 'i' solo es visible
   {                       // dentro del bucle for
      i++
   }                       // 'f' es visible solo
   float f;                // dentro de loop()
}
```

tipos de datos

byte

Byte almacena un valor numérico de 8 bits sin decimales. Tienen un rango entre 0 y 255.

```
byte unaVariable = 180;    // declara 'unaVariable' como
                           // de tipo byte
```

int

Enteros son un tipo de datos primarios que almacenan valores numéricos de 16 bits sin decimales comprendidos en el rango 32,767 to -32,768.

```
int unaVariable = 1500;    // declara 'unaVariable' como
                           // una variable de tipo entero
```

Nota: Las variables de tipo entero “int” pueden sobrepasar su valor máximo o mínimo como consecuencia de una operación. Por ejemplo, si x = 32767 y una posterior declaración agrega 1 a x, x = x + 1 entonces el valor se x pasará a ser -32.768. (algo así como que el valor da la vuelta).

long

El formato de variable numérica de tipo extendido “long” se refiere a números enteros (tipo 32 bits) sin decimales que se encuentran dentro del rango -2147483648 a 2147483647.

```
long unaVariable = 90000;  // declara 'unaVariable' como
                           // de tipo long
```

float

El formato de dato del tipo “punto flotante” “float” se aplica a los números con decimales. Los números de punto flotante tienen una mayor resolución que los de 32 bits con un rango comprendido 3.4028235E +38 a +38-3.4028235E.

```
float unaVariable = 3.14;  // declara 'unaVariable' como
                           // de tipo flotante
```

Nota: Los números de punto flotante no son exactos, y pueden producir resultados extraños en las comparaciones. Los cálculos matemáticos de punto flotante son también mucho más lentos que los del tipo de números enteros, por lo que debe evitarse su uso si es posible.

arrays

Un array es un conjunto de valores a los que se accede con un número índice. Cualquier valor puede ser recogido haciendo uso del nombre de la matriz y el número del índice. El primer valor de la matriz es el que está indicado con el índice 0, es decir el primer valor del conjunto es el de la posición 0. Un array tiene que ser declarado y opcionalmente asignados valores a cada posición antes de ser utilizado.

```
int miArray[] = {valor0, valor1, valor2...}
```

Del mismo modo es posible declarar una matriz indicando el tipo de datos y el tamaño y posteriormente, asignar valores a una posición especifica:

```
int miArray[5];      // declara un array de enteros de 6
                     // posiciones
miArray[3] = 10;     // asigna l valor 10 a la posición 4
```

Para leer de un array basta con escribir el nombre y la posición a leer:

```
x = miArray[3];      // x ahora es igual a 10 que está en
                     // la posición 3 del array
```

Las matrices se utilizan a menudo para estamentos de tipo bucle, en los que la variable de incremento del contador del bucle se utiliza como índice o puntero del array. El siguiente ejemplo usa una matriz para el parpadeo de un LED.

Utilizando un bucle tipo for, el contador comienza en cero 0 y escribe el valor que figura en la posición de índice 0 en la serie que hemos escrito dentro del array parpadeo[], en este caso 180, que se envía a la salida analógica tipo PWM configurada en el PIN10, se hace una pausa de 200 ms y a continuación se pasa al siguiente valor que asigna el índice “i”.

```
int ledPin = 10;           // LED en el PIN 10
byte parpadeo[] = {180, 30, 255, 200, 10, 90, 150, 60};
                           //  array de 8 valores
void setup()
{
   pinMode(ledPin, OUTPUT);       // configura la salida
}

void loop()
{
   for(int i=0; i<7; i++)
   {
      analogWrite(ledPin, parpadeo[i]);
      delay(200); // espera 200ms
   }
}
```

aritmética

Los operadores aritméticos que se incluyen en el entorno de programación son suma, resta, multiplicación y división. Estos devuelven la suma, diferencia, producto, o cociente (respectivamente) de dos operandos.

```
y = y + 3;  x = x - 7;   i = j * 6;   r = r / 5;
```

La operación se efectúa teniendo en cuenta el tipo de datos que hemos definido para los operandos (int, dbl, float, etc..), por lo que, por ejemplo, si definimos 9 y 4 como enteros “int”, 9 / 4 devuelve de resultado 2 en lugar de 2,25 ya que el 9 y 4 se valores de tipo entero “int” (enteros) y no se reconocen los decimales con este tipo de datos.

Esto también significa que la operación puede sufrir un desbordamiento si el resultado es más grande que lo que puede ser almacenada en el tipo de datos. Recordemos el alcance de los tipos de datos numéricos que ya hemos explicado anteriormente.

Si los operandos son de diferentes tipos, para el cálculo se utilizará el tipo más grande de los operandos en juego. Por ejemplo, si uno de los números (operandos) es del tipo float y otra de tipo integer, para el cálculo se utilizará el método de float es decir el método de coma flotante.

Elija el tamaño de las variables de tal manera que sea lo suficientemente grande como para que los resultados sean lo precisos que usted desea. Para las operaciones que requieran decimales utilice variables tipo float, pero sea consciente de que las operaciones con este tipo de variables son más lentas a la hora de realizarse el computo.

Nota: Utilice el operador (int) para convertir un tipo de variable a otro sobre la marcha. Por ejemplo, i = (int) 3,6 establecerá i igual a 3.

asignaciones compuestas

Las asignaciones compuestas combinan una operación aritmética con una variable asignada. Estas son comúnmente utilizadas en los bucles tal como se describe más adelante. Estas asignaciones compuestas pueden ser:

```
x ++   // igual que x = x +1, o incremento de x en +1
x --   // igual que x = x - 1, o decremento de x en -1
x += y // igual que x = x + y, o incremento de x en +y
x -= y // igual que x = x - y, o decremento de x en -y
x *= y // igual que x = x * y, o multiplica x por y
x /= y // igual que x = x / y, o divide x por y
```

Nota: Por ejemplo, x * = 3 hace que x se convierta en el triple del antiguo valor x y por lo tanto x es reasignada al nuevo valor.

operadores de comparación

Las comparaciones de una variable o constante con otra se utilizan con frecuencia en las estructuras condicionales del tipo if.. para testear si una condición es verdadera. En los ejemplos que siguen en las próximas páginas se verá su utilización práctica usando los siguientes tipo de condicionales:

```
x == y       // x es igual a y
x != y       // x no es igual a y
x < y        // x es menor que y
x > y        // x es mayor que y
x <= y       // x es menor o igual que y
x >= y       // x es mayor o igual que y
```

operadores lógicos

Los operadores lógicos son usualmente una forma de comparar dos expresiones y devolver un VERDADERO o FALSO dependiendo del operador. Existen tres operadores lógicos, AND (&&), OR (||) y NOT (!), que a menudo se utilizan en estamentos de tipo if:

Logica AND:

```
if (x > 0 && x < 5) // cierto sólo si las dos
expresiones               // son ciertas
```

Logica OR:

```
if (x > 0 || y > 0) // cierto si una cualquiera de las
                          // expresiones es cierta
```

Logica NOT:

```
if (!x > 0)         // cierto solo si la expresión es
                          // falsa
```

constantes

El lenguaje de programación de Arduino tiene unos valores predeterminados, que son llamados constantes. Se utilizan para hacer los programas más fáciles de leer. Las constantes se clasifican en grupos.

cierto/falso (true/false)

Estas son constantes booleanas que definen los niveles HIGH (alto) y LOW (bajo) cuando estos se refieren al estado de las salidas digitales. FALSE se asocia con 0 (cero), mientras que TRUE se asocia con 1, pero TRUE también puede ser cualquier otra cosa excepto cero. Por lo tanto, en sentido booleano, -1, 2 y -200 son todos también se define como TRUE. (esto es importante tenerlo en cuanta).

```
if (b == TRUE);
{
ejecutar las instrucciones;
}
```

high/low

Estas constantes definen los niveles de salida altos o bajos y se utilizan para la lectura o la escritura digital para las patillas. ALTO se define como en la lógica de nivel 1, ON, ó 5 voltios, mientras que BAJO es lógica nivel 0, OFF, o 0 voltios.

```
digitalWrite(13, HIGH);    // activa la salida 13 con un
                           // nivel alto (5v.)
```

input/output

Estas constantes son utilizadas para definir, al comienzo del programa, el modo de funcionamiento de los pines mediante la instrucción pinMode de tal manera que el pin puede ser una entrada INPUT o una salida OUTPUT.

```
pinMode(13, OUTPUT);       // designamos que el PIN 13 es
                           // una salida
```

control de fjujo

if (si condicional)

if es un estamento que se utiliza para probar si una determinada condición se ha alcanzado, como por ejemplo averiguar si un valor analógico está por encima de un cierto número, y ejecutar una serie de declaraciones (operaciones) que se escriben dentro de llaves, si es verdad. Si es falso (la condición no se cumple) el programa salta y no ejecuta las operaciones que están dentro de las llaves, El formato para if es el siguiente:

```
if (unaVariable ?? valor)
{
ejecutaInstrucciones;
}
```

En el ejemplo anterior se compara una variable con un valor, el cual puede ser una variable o constante. Si la comparación, o la condición entre paréntesis se cumple (es cierta), las declaraciones dentro de los corchetes se ejecutan. Si no es así, el programa salta sobre ellas y sigue.

Nota: Tenga en cuenta el uso especial del símbolo '=', poner dentro de if (x = 10), podría parecer que es valido pero sin embargo no lo es ya que esa expresión asigna el valor 10 a la variable x, por eso dentro de la estructura if se utilizaría X==10 que en este caso lo que hace el programa es comprobar si el valor de x es 10.. Ambas cosas son distintas por lo tanto dentro de las estructuras if, cuando se pregunte por un valor se
debe poner el signo doble de igual “==”.

if… else (si….. sino ..)

if… else viene a ser un estructura que se ejecuta en respuesta a la idea “si esto no se cumple haz esto otro”. Por ejemplo, si se desea probar una entrada digital, y hacer una cosa si la entrada fue alto o hacer otra cosa si la entrada es baja, usted escribiría que de esta manera:

```
if (inputPin == HIGH)
  {
  instruccionesA;
  }
else
  {
  instruccionesB;
  }
```

Else puede ir precedido de otra condición de manera que se pueden establecer varias estructuras condicionales de tipo unas dentro de las otras (anidamiento) de forma que sean mutuamente excluyentes pudiéndose ejecutar a la vez. Es incluso posible tener un número ilimitado de estos condicionales. Recuerde sin embargo qué sólo un conjunto de declaraciones se llevará a cabo dependiendo de la condición probada:

```
if (inputPin < 500)
   {
   instruccionesA;
   }
else if (inputPin >= 1000)
   {
   instruccionesB;
   {
else
   {
   instruccionesC;
   }
```

Nota: Un estamento de tipo if prueba simplemente si la condición dentro del paréntesis es verdadera o falsa. Esta declaración puede ser cualquier declaración válida. En el anterior ejemplo, si cambiamos y ponemos (inputPin == HIGH). En este caso, el

estamento if sólo chequearía si la entrada especificado esta en nivel alto (HIGH), o +5v.

for

La declaración for se usa para repetir un bloque de sentencias encerradas entre llaves un número determinado de veces. Cada vez que se ejecutan las instrucciones del bucle se vuelve a testear la condición. La declaración for tiene tres partes separadas por (;), vemos el ejemplo de su sintaxis:

```
for (inicialización; condición; expresión)
  {
  Instrucciones;
  }
```

La inicialización de una variable local se produce una sola vez y la condición se testea cada vez que se termina la ejecución de las instrucciones dentro del bucle. Si la condición sigue cumpliéndose, las instrucciones del bucle se vuelven a ejecutar. Cuando la condición no se cumple, el bucle termina.

El siguiente ejemplo inicia el entero i en el 0, y la condición es probar que el valor es inferior a 20 y si es cierto i se incrementa en 1 y se vuelven a ejecutar las instrucciones que hay dentro de las llaves:

```
for (int i=0; i<20; i++)   // declara i y prueba si es
{                          // menor que 20, incrementa i.
digitalWrite(13, HIGH);    // enciende el pin 13
delay(250);                // espera ¼ seg.
digitalWrite(13, LOW);     // apaga el pin 13
delay(250);                // espera ¼ de seg.
}
```

Nota: El bucle en el lenguaje C es mucho más flexible que otros bucles encontrados en algunos otros lenguajes de programación, incluyendo BASIC. Cualquiera de los tres elementos de cabecera puede omitirse, aunque el punto y coma es obligatorio. También las declaraciones de inicialización, condición y expresión puede ser cualquier estamento válido en lenguaje C sin relación con las variables declaradas. Estos tipos de estados son raros pero

permiten disponer soluciones a algunos problemas de programación raras.

while

Un bucle del tipo while es un bucle de ejecución continua mientras se cumpla la expresión colocada entre paréntesis en la cabecera del bucle. La variable de prueba tendrá que cambiar para salir del bucle. La situación podrá cambiar a expensas de una expresión dentro el código del bucle o también por el cambio de un valor en una entrada de un sensor.

```
while (unaVariable ?? valor)
{
ejecutarSentencias;
}
```

El siguiente ejemplo testea si la variable "unaVariable" es inferior a 200 y, si es verdad, ejecuta las declaraciones dentro de los corchetes y continuará ejecutando el bucle hasta que 'unaVariable' no sea inferior a 200.

```
While (unaVariable < 200)  // testea si es menor que 200
{
instrucciones;             // ejecuta las instrucciones
                           // entre llaves
unaVariable++;             // incrementa la variable en 1
}
```

do… while

El bucle do while funciona de la misma manera que el bucle while, con la salvedad de que la condición se prueba al final del bucle, por lo que el bucle siempre se ejecutará al menos una vez.

```
do
{
Instrucciones;
} while (unaVariable ?? valor);
```

El siguiente ejemplo asigna el valor leído leeSensor() a la variable 'x', espera 50 milisegundos, y luego continua mientras que el valor de la 'x' sea inferior a 100:

```
do
{
```

```
x = leeSensor();
delay(50);
} while (x < 100);
```

e/s digitales

pinMode(pin, mode)

Esta instrucción es utilizada en la parte de configuración setup () y sirve para configurar el modo de trabajo de un PIN pudiendo ser INPUT (entrada) u OUTPUT (salida).

```
pinMode(pin, OUTPUT);      // configura 'pin' como salida
```

Los terminales de Arduino, por defecto, están configurados como entradas, por lo tanto no es necesario definirlos en el caso de que vayan a trabajar como entradas. Los pines configurados como entrada quedan, bajo el punto de vista eléctrico, como entradas en estado de alta impedancia.

Estos pines tienen a nivel interno una resistencia de 20 KΩ a las que se puede acceder mediante software. Estas resistencias se accede de la siguiente manera:

```
pinMode(pin, INPUT);       // configura el 'pin' como
                           // entrada
digitalWrite(pin, HIGH);   // activa las resistencias
                           // internas
```

Las resistencias internas normalmente se utilizan para conectar las entradas a interruptores. En el ejemplo anterior no se trata de convertir un pin en salida, es simplemente un método para activar las resistencias interiores.

Los pins configurado como OUTPUT (salida) se dice que están en un estado de baja impedancia estado y pueden proporcionar 40 mA (miliamperios) de corriente a otros dispositivos y circuitos. Esta corriente es suficiente para alimentar un diodo LED (no olvidando poner una resistencia en serie), pero no es lo suficiente grande como para alimentar cargas de mayor consumo como relés, solenoides, o motores.

Un cortocircuito en las patillas Arduino provocará una corriente elevada que puede dañar o destruir el chip Atmega. A menudo es

una buena idea conectar en la OUTUPT (salida) una resistencia externa de 470 o de 1000 Ω.

digitalRead(pin)

Lee el valor de un pin (definido como digital) dando un resultado HIGH (alto) o LOW (bajo). El pin se puede especificar ya sea como una variable o una constante (0-13).

```
valor = digitalRead(Pin);  // hace que 'valor sea igual
                           // al estado leído en ´Pin´
```

digitalWrite(pin, value)

Envía al ´pin´ definido previamente como OUTPUT el valor HIGH o LOW (poniendo en 1 o 0 la salida). El pin se puede especificar ya sea como una variable o como una constante (0-13).

```
digitalWrite(pin, HIGH);   // deposita en el 'pin' un
valor                              // HIGH (alto o 1)
```

El siguiente ejemplo lee el estado de un pulsador conectado a una entrada digital y lo escribe en el ´pin´de salida LED:

```
int led      = 13;  // asigna a LED el valor 13
int boton = 7;      // asigna a botón el valor 7
int valor = 0;      // define el valor y le asigna el
                    // valor 0

void setup()
{
pinMode(led, OUTPUT); // configura el led (pin13) como
salida
pinMode(boton, INPUT); // configura botón (pin7) como
entrada
}
void loop()
{
valor = digitalRead(boton); //lee el estado de la
                           // entrada botón
digitalWrite(led, valor); // envía a la salida ´led´el
}                          // valor leído
```

e/s analógicas

analogRead(pin)

Lee el valor de un determinado pin definido como entrada analógica con una resolución
de 10 bits. Esta instrucción sólo funciona en los pines (0-5). El rango de valor que podemos leer oscila de 0 a 1023.

```
valor = analogRead(pin);   // asigna a valor lo que lee
                           // en la entrada ´pin'
```

Nota: Los pins analógicos (0-5) a diferencia de los pines digitales, no necesitan ser declarados como INPUT u OUPUT ya que son siempre INPUT´s.

analogWrite(pin, value)

Esta instrucción sirve para escribir un pseudo-valor analógico utilizando el procedimiento de modulación por ancho de pulso (PWM) a uno de los pin´s de Arduino marcados como “pin PWM”. El más reciente Arduino, que implementa el chip ATmega168, permite habilitar como salidas analógicas tipo PWM los pines 3, 5, 6, 9, 10 y 11. Los modelos de Arduino más antiguos que implementan el chip ATmega8, solo tiene habilitadas para esta función los pines 9, 10 y 11. El valor que se puede enviar a estos pines de salida analógica puede darse en forma de variable o constante, pero siempre con un margen de 0-255.

```
analogWrite(pin, valor);   // escribe 'valor' en el 'pin'
                           // definido como analógico
```

Si enviamos el valor 0 genera una salida de 0 voltios en el pin especificado; un valor de 255 genera una salida de 5 voltios de salida en el pin especificado. Para valores de entre 0 y 255, el pin saca tensiones entre 0 y 5 voltios - el valor HIGH de salida equivale a 5v (5 voltios). Teniendo en cuenta el concepto de señal PWM , por ejemplo, un valor de 64 equivaldrá a mantener 0 voltios de tres cuartas partes del tiempo y 5 voltios a una cuarta parte del tiempo; un valor de 128 equivaldrá a mantener la salida en 0 la mitad del

tiempo y 5 voltios la otra mitad del tiempo, y un valor de 192 equivaldrá a mantener en la salida 0 voltios una cuarta parte del tiempo y de 5 voltios de tres cuartas partes del tiempo restante.

Debido a que esta es una función de hardware, en el pin de salida analógica (PWN) se generará una onda constante después de ejecutada la instrucción analogWrite hasta que se llegue a ejecutar otra instrucción analogWrite (o una llamada a digitalRead o digitalWrite en el mismo pin).

Nota: Las salidas analógicas a diferencia de las digitales, no necesitan ser declaradas como INPUT u OUTPUT..

El siguiente ejemplo lee un valor analógico de un pin de entrada analógica, convierte el valor dividiéndolo por 4, y envía el nuevo valor convertido a una salida del tipo PWM o salida analógica:

```
int led = 10;        // define el pin 10 como ´led´
int analog = 0;      // define el pin 0 como ´analog´
int valor;           // define la variable ´valor´

void setup(){}       // no es necesario configurar
                     // entradas y salidas

void loop()
{
valor = analogRead(analog);       // lee el pin 0 y lo
asocia a                          //la variable valor
valor /= 4;                //divide valor entre 4 y lo
                                  //reasigna a valor
analogWrite(led, value);   // escribe en el pin10 valor
}
```

control del tiempo

delay(ms)

Detiene la ejecución del programa la cantidad de tiempo en ms que se indica en la propia instrucción. De tal manera que 1000 equivale a 1seg.

```
delay(1000); // espera 1 segundo
```

millis()

Devuelve el número de milisegundos transcurrido desde el inicio del programa en Arduino hasta el momento actual. Normalmente será un valor grande (dependiendo del tiempo que este en marcha la aplicación después de cargada o después de la última vez que se pulsó el botón “reset” de la tarjeta).

```
valor = millis();   // valor recoge el número de
                    // milisegundos
```

Nota: Este número se desbordará (si no se resetea de nuevo a cero), después de aproximadamente 9 horas.

Matemáticas

min(x, y)

Calcula el mínimo de dos números para cualquier tipo de datos devolviendo el número más pequeño.

```
valor = min(valor, 100);   // asigna a valor el más
                           // pequeños  de los dos
                           // números especificados.
```

Si 'valor' es menor que 100 valor recogerá su propio valor si ´valor´ es mayor que 100 valor pasara a valer 100.

max(x, y)

Calcula el máximo de dos números para cualquier tipo de datos devolviendo el número mayor de los dos.

```
valor = max(valor, 100);   // asigna a valor el mayor de
                           // los dos números 'valor' y
                           // 100.
```

De esta manera nos aseguramos de que valor será como mínimo 100.

aleatorios

randomSeed(seed)

Establece un valor, o semilla, como punto de partida para la función random().

```
randomSeed(valor);  // hace que valor sea la semilla del
                    // random
```

Debido a que Arduino es incapaz de crear un verdadero número aleatorio, randomSeed le permite colocar una variable, constante, u otra función de control dentro de la función random, lo que permite generar números aleatorios "al azar". Hay una variedad de semillas, o funciones, que pueden ser utilizados en esta función, incluido millis () o incluso analogRead () que permite leer ruido eléctrico a través de un pin analógico.

random(max), random(min, max)

La función random devuelve un número aleatorio entero de un intervalo de valores especificado entre los valores min y max.

```
valor = random(100, 200);  // asigna a la variable
                           // 'valor' un numero aleatorio
                           // comprendido entre 100-200
```

Nota: Use esta función después de usar el randomSeed().

El siguiente ejemplo genera un valor aleatorio entre 0-255 y lo envía a una salida analógica PWM :

```
int randNumber;     // variable que almacena el valor
                    // aleatorio
int led = 10;       // define led como 10

void setup() {}     // no es necesario configurar nada

void loop()
{
```

```
randomSeed(millis());      // genera una semilla para
                           // aleatorio a partir
                           // de la función millis()

randNumber = random(255);  // genera número aleatorio
                           // entre 0-255
analogWrite(led, randNumber);     // envía a la salida
                                  // led de tipo PWM el
                                  // valor
delay(500);   // espera 0,5 seg.
}
```

comunicación serie

Serial.begin(rate)

Abre el puerto serie y fija la velocidad en baudios para la transmisión de datos en serie.
El valor típico de velocidad para comunicarse con el ordenador es 9600, aunque otras velocidades pueden ser soportadas.

```
void setup()
                              {
Serial.begin(9600); // abre el Puerto serie
}      // configurando la velocidad en 9600 bps
```

Nota: Cuando se utiliza la comunicación serie los pins digital 0 (RX) y 1 (TX) no puede utilizarse al mismo tiempo.

Serial.println(data)

Imprime los datos en el puerto serie, seguido por un retorno de carro automático y salto de línea. Este comando toma la misma forma que Serial.print(), pero es más fácil para la lectura de los datos en el Monitor Serie del software.

```
Serial.println(analogValue);      // envía el valor
                                  // 'analogValue' al
                                  // puerto
```

Nota: Para obtener más información sobre las distintas posibilidades de Serial.println () y Serial.print () puede consultarse el sitio web de Arduino.

El siguiente ejemplo toma de una lectura analógica pin0 y envía estos datos al ordenador cada 1 segundo.

```
void setup()
{
Serial.begin(9600); // configura el puerto serie a
                    // 9600bps
}
void loop()
```

```
{
Serial.println(analogRead(0)); // envía valor analógico
delay(1000);  // espera 1 segundo
}
```

Serial.print(data, data type)

Vuelca o envía un número o una cadena de carateres, al puerto serie. Dicho comando puede tomar diferentes formas, dependiendo de los parámetros que utilicemos para definir el formato de volcado de los números.

Parámetros

data: el número o la cadena de caracteres a volcar o enviar.

data type: determina el formato de salida de los valores numéricos (decimal, octal, binario, etc...) DEC, OCT, BIN, HEX, BYTE , si no se pe nada vuelva ASCII

Ejemplos:

```
Serial.print(b)      // Vuelca o envía el valor de b como
                     // un número decimal en caracteres
                     // ASCII.
int b = 79;
Serial.print(b); // imprime la cadena "79".
Serial.print(b, HEX);      // Vuelca o envía el valor de
                           // b como un número hexdecimal
                           // en caracteres ASCII "4F".
Serial.print(b, OCT);      // Vuelca o envía el valor de
                           // b como un número Octal en
                           // caracteres ASCII  "117".
Serial.print(b, BIN)       // Vuelca o envía el valor de
                           // b como un número binario en
                           // caracteres ASCII "1001111".
Serial.print(b, BYTE);     // Devuelve el caracter "O",
                           // el cual representa el
                           // caracter ASCII del valor
                           // 79. (Ver tabla ASCII).
Serial.print(str);  //Vuelca o envía la cadena de
                    // caracteres como una cadena ASCII.
Serial.print("Hello World!"); // vuelca "Hello World!".
```

Serial.avaible()

Devuelve Un entero con el número de bytes disponibles para leer desde el buffer serie, o 0 si no hay ninguno. Si hay algún dato disponible, SerialAvailable() será mayor que 0. El buffer serie puede almacenar como máximo 64 bytes.

```
int Serial.available()     // Obtiene un número entero
                           // con el número de bytes
                           // (caracteres) disponibles
                           // para leer o capturar desde
                           // el puerto serie
```

Ejemplo

```
int incomingByte = 0; // almacena el dato serie

void setup() {
Serial.begin(9600); // abre el puerto serie, y le asigna
                    // la velocidad de 9600 bps
}
void loop() {
if (Serial.available() > 0)       // envía datos sólo si
{                                 // los recibe:
incomingByte = Serial.read();// lee el byte de entrada:
                                  //lo vuelca a pantalla
Serial.print("I received: ");
Serial.println(incomingByte, DEC);
}
}
```

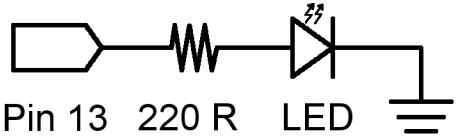
Serial.Read()

Lee o captura un byte (un caracter) desde el puerto serie.
Devuelve el siguiente byte (carácter) desde el puerto serie, o -1 si no hay ninguno.
Ejemplo:

```
int incomingByte = 0; // almacenar el dato serie

void setup() {
Serial.begin(9600); // abre el puerto serie,y le asigna
                    // la velocidad de 9600 bps
}

void loop() {
if (Serial.available() > 0)// envía datos sólo si los
{                          // recibe
incomingByte = Serial.read();     // lee el byte de
                                  // entrada y lo vuelca
Serial.print("I received: ");     // a pantalla
Serial.println(incomingByte, DEC);
}
}
```

apéndices

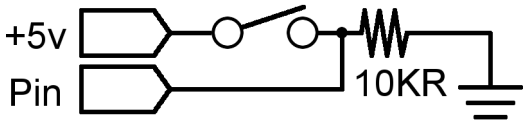
salida digital



Éste es el ejemplo básico equivalente al "hola mundo" de cualquier lenguaje de programación haciendo simplemente el encendido y apagado de un led. En este ejemplo el LED está conectado en el pin13, y se enciende y apaga “parpadea” cada segundo. La resistencia que se debe colocar en serie con el led en este caso puede omitirse ya que el pin13 de Arduino ya incluye en la tarjeta esta resistencia,

```
int ledPin = 13; // LED en el pin digital 13

void setup()  // configura el pin de salida


{
pinMode(ledPin, OUTPUT); // configura el pin 13 como
salida
}
void loop()   // inicia el bucle del programa
{
digitalWrite(ledPin, HIGH); // activa el LED
delay(1000); // espera 1 segundo
digitalWrite(ledPin, LOW); // desactiva el LED


delay(1000); // espera 1 segundo
}
```

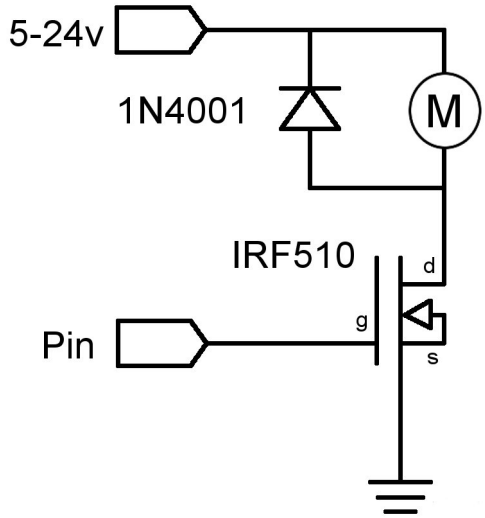
entrada digital



Ésta es la forma más sencilla de entrada con sólo dos posibles estados: encendido o apagado. En este ejemplo se lee un simple switch o pulsador conectado a PIN2. Cuando el interruptor está cerrado el pin de entrada se lee ALTO y encenderá un LED colocado en el PIN13

```
int ledPin = 13;     // pin 13 asignado para el LED de
                     // salida
int inPin = 2;       // pin 2 asignado para el pulsador


void setup()         // Configura entradas y salidas
{
 pinMode(ledPin, OUTPUT);  // declara LED como salida
 pinMode(inPin, INPUT);    // declara pulsador como
}                          // entrada

void loop()
{
if (digitalRead(inPin) == HIGH)   // testea si la entrada
{                                 // esta activa HIGH
 digitalWrite(ledPin, HIGH);      // enciende el LED
 delay(1000);               // espera 1 segundo
 digitalWrite(ledPin, LOW);       // apaga el LED
}
}
```

salida de alta corriente de consumo

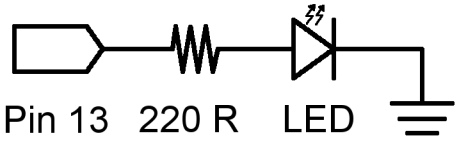


A veces es necesario controlar cargas de más de los 40 mA que es capaz de suministrar la tarjeta Arduino. En este caso se hace uso de un transistor MOSFET que puede alimentar cargas de mayor consumo de corriente. El siguiente ejemplo muestra como el transistor MOSFET conmuta 5 veces cada segundo.

Nota: El esquema muestra un motor con un diodo de protección por ser una carga inductiva. En los casos que las cargas no sean inductivas no será necesario colocar el diodo.

```
int outPin = 5;     // pin de salida para el MOSFET

void setup()
{
  pinMode(outPin, OUTPUT); // pin5 como salida
}

void loop()
{
  for (int i=0; i<=5; i++) // repetir bucle 5 veces
  {
    digitalWrite(outPin, HIGH);   // activa el MOSFET
    delay(250);                   // espera 1/4 segundo
    digitalWrite(outPin, LOW);    // desactiva el MOSFET
    delay(250);                   // espera 1/4 segundo
  }
  delay(1000); // espera 1 segundo
}
```

salida analógica del tipo pwm

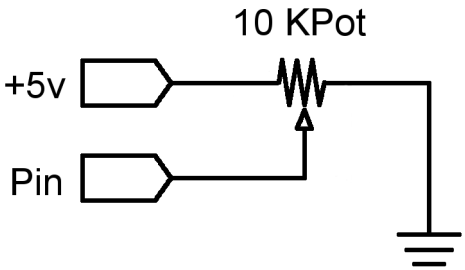


La Modulación de Impulsos en Frecuencia (PWM) es una forma de conseguir una “falsa” salida analógica. Esto podría ser utilizado para modificar el brillo de un LED o controlar un servo motor. El siguiente ejemplo lentamente hace que el LED se ilumine y se apague haciendo uso de dos bucles.

```
int ledPin = 9;      // pin PWM para el LED


void setup(){} // no es necesario configurar nada

void loop()
{
  for (int i=0; i<=255; i++) // el valor de i asciende
  {
  analogWrite(ledPin, i); // se escribe el valor de i en
                          // el PIN de salida del LED
  delay(100);             // pauses for 100ms
  }
  for (int i=255; i>=0; i--) // el valor de i desciende
  {
  analogWrite(ledPin, i);  // se escribe el valor de ii
  delay(100);              // pasusa durante 100ms
  }
}
```

entrada con potenciómetro

(entrada analógica)

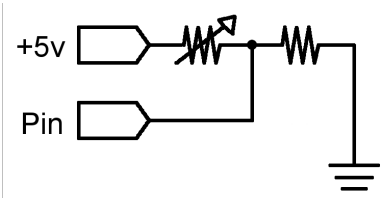


El uso de un potenciómetro y uno de los pines de entrada analógica-digital de Arduino (ADC) permite leer valores analógicos que se convertirán en valores dentro del rango de 0-1024. El siguiente ejemplo utiliza un potenciómetro para controlar un el tiempo de parpadeo de un LED.

```
int potPin = 0;      // pin entrada para potenciómetro
int ledPin = 13;     // pin de salida para el LED

void setup()
{
  pinMode(ledPin, OUTPUT); // declara ledPin como SALIDA
}

void loop()
{
  digitalWrite(ledPin, HIGH);     // pone ledPin en on
  delay(analogRead(potPin));      // detiene la ejecución
                                  // un tiempo "potPin"
  digitalWrite(ledPin, LOW);      // pone ledPin en off
  delay(analogRead(potPin));      // detiene la ejecución
                                  // un tiempo "potPin"
}
```

entrada conectada a resistencia variable

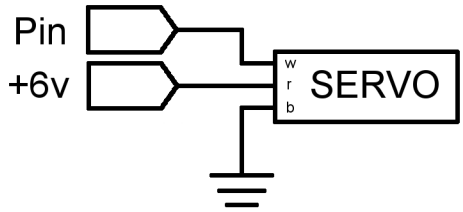
(entrada analógica)



Las resistencias variables como los sensores de luz LCD los termistores, sensores de esfuerzos, etc, se conectan a las entradas analógicas para recoger valores de parámetros físicos. Este ejemplo hace uso de una función para leer el valor analógico y establecer un tiempo de retardo. Este tiempo controla el brillo de un diodo LED conectado en la salida.

```
int ledPin =   9;      // Salida analógica PWM para
                       //conectar a LED
int analogPin = 0;     // resistencia variable conectada a la
                       // entrada analógica pin 0
void setup(){} // no es necesario configurar entradas y salidas


void loop()
{
  for (int i=0; i<=255; i++) // incremento de valor de i
   {
    analogWrite(ledPin, i);   // configura el nivel brillo con
                              // el valor de i
    delay(delayVal());        // espera un tiempo
   }
  for (int i=255; i>=0; i--) // decrementa el valor de i
  {
   analogWrite(ledPin, i);    // configura el nivel de brillo
                              // con el valor de i
   delay(delayVal());         // espera un tiempo
  }
}
int delayVal()
{
  int v;                      // crea una variable temporal
                              // (local)
  v = analogRead(analogPin);  // lee valor analógico
  v /= 8;                     // convierte el valor leído de 0-
                              // 1024 a 0-128
  return v; // devuelve el valor v
}
```

salida conectada a servo



Los servos de modelismo tienen un motor y unos engranajes cuya salida se puede mover en un arco de 180 º y contienen la electrónica necesaria para ello. Todo lo que se necesita es un pulso enviado cada 20ms. Este ejemplo utiliza la función servoPulse para mover el servo de
10º a 170 º.

```
int servoPin = 2;   // servo conectado al pin digital 2
int myAngle;        // ángulo del servo de 0-180
int pulseWidth;     // anchura del pulso para la función
                    // servoPulse
void setup()
{
  pinMode(servoPin, OUTPUT);      // configura pin 2 como
}                                 // salida

void servoPulse(int servoPin, int myAngle)
{
  pulseWidth = (myAngle * 10) + 600;    // determina
                                        // retardo
  digitalWrite(servoPin, HIGH);   // activa el servo
  delayMicroseconds(pulseWidth); // pausa
  digitalWrite(servoPin, LOW);    // desactiva el servo
  delay(20);                      // retardo de refresco
}
void loop()  // el servo inicia su recorrido en 10° y
{            // gira hasta 170°
  for (myAngle=10; myAngle<=170; myAngle++)
  {
    servoPulse(servoPin, myAngle);
  }
            // el servo vuelve desde 170° hasta 10°
  for (myAngle=170; myAngle>=10; myAngle--)
  {
    servoPulse(servoPin, myAngle);
  }
}
```

B2. Manual d'ús de Fritzing

FRITZING

CONSTRUYENDO UN CIRCUITO

Venezuela, 2012
Germana Oliveira
germanaoliveirab@gmail.com

El siguiente es una traducción de la guía de aprendizaje "Building a Circuit" de la sección Learning encontrada en en la página oficial de fritzing.

Introducción

En este pequeño tutorial construiremos un circuito y aprenderemos las características básicas del entorno de Fritzing.

Tabla de contenido.

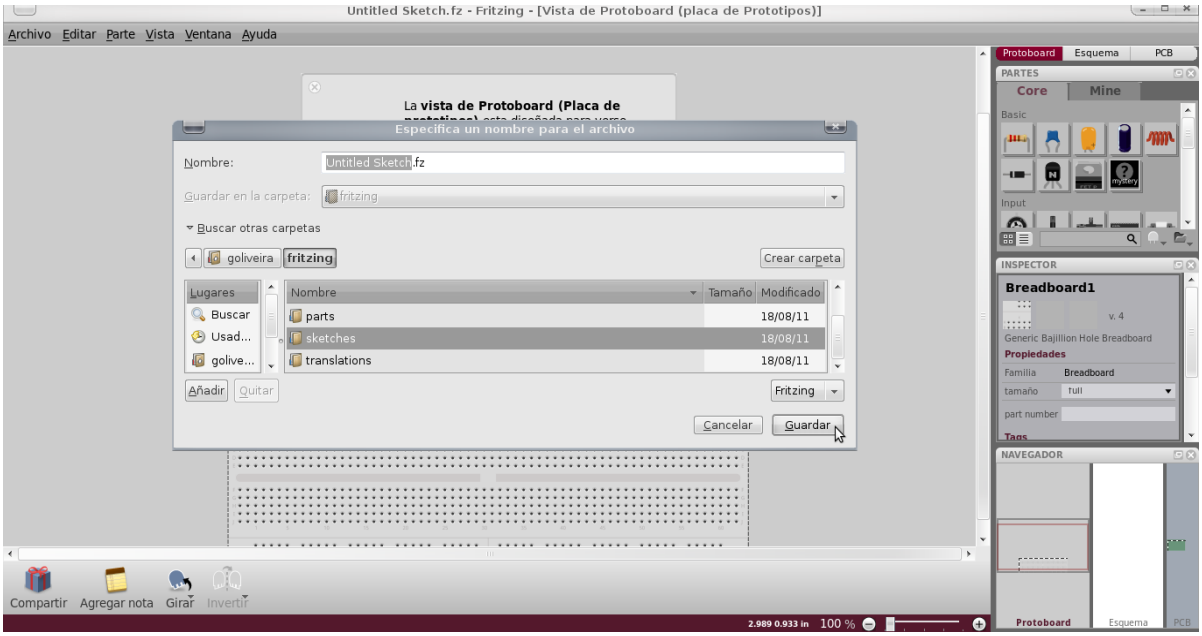
Comenzar un nuevo proyecto

Antes de comenzar un proyecto en Fritzing, debes construir el circuito electrónico en el mundo real y asegurarte que funcione correctamente. Luego podrás reconstruir el circuito virtualmente usando Fritzing.

Comencemos ejecutando Fritzing, nombrando y guardando nuestro proyecto. Es recomendable guardar el proyecto al inicio y de vez en cuando durante el desarrollo del mismo, ya que Fritzing aún esta en su versión Alfa y puede, a veces, fallar inesperadamente.

1. Desde la barra de menú de Fritzing, selecciona Archivo > Guardar como ...
2. Coloca un nombre para el proyecto y un lugar donde guardarlo y haz click en Guardar.

Imagen No.1
Guardar como

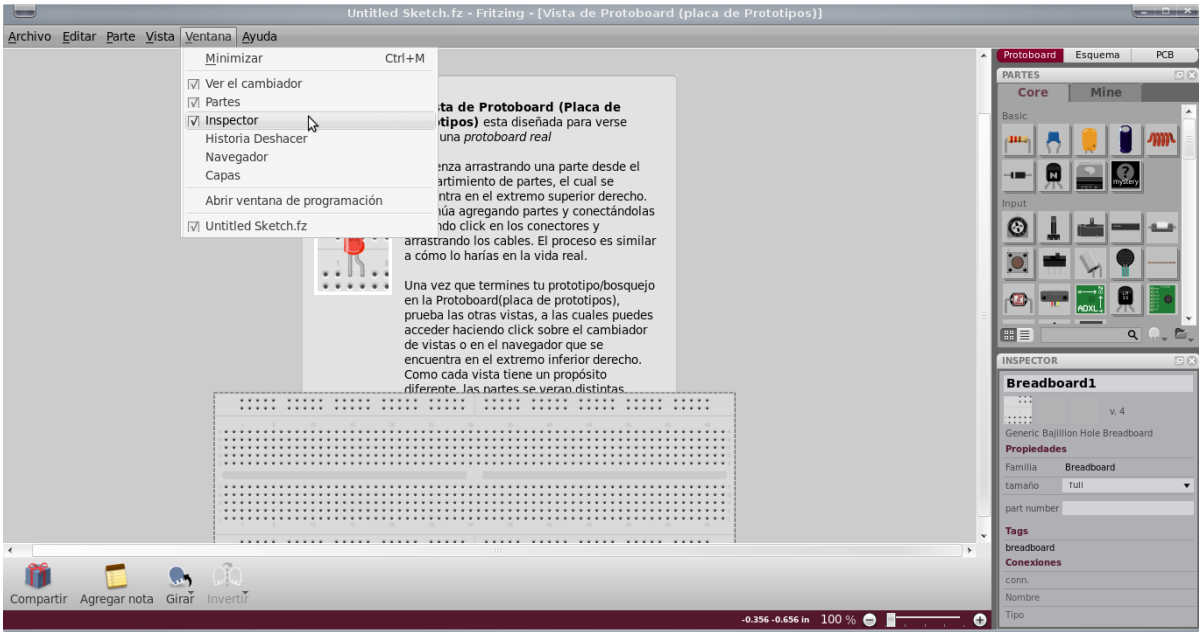


Organización del entorno

Antes de comenzar a trabajar, puede ser que queramos arreglar el entorno en base a nuestros gustos o necesidades.

1. Desde la barra de menú de Fritzing, selecciona Ventana > y selecciona (observa el símbolo de “check” al seleccionar) la paleta que quisieras ver en el entorno.
2. Arrastra y suelta la ventana de la paleta seleccionada en cualquier lugar dentro del entorno, y observa como las ventanas se reajustan, se combinan y flotan.
3. Selecciona la vista de placa en el Navegador, si no esta ya seleccionada.

Imagen No.2
Mostrar ventanas

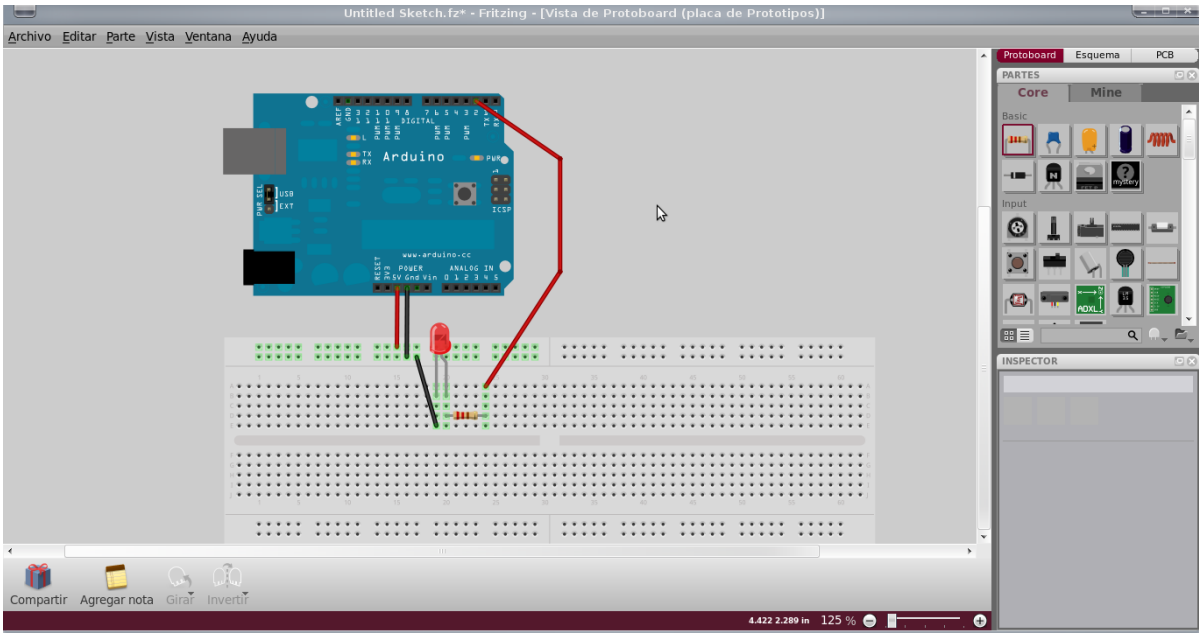


Construyendo un circuito

Asegura te que el circuito funcione adecuadamente en el mundo real, luego construye nuevamente el circuito, esta vez, virtualmente usando Fritzing, siguiendo las siguientes instrucciones:

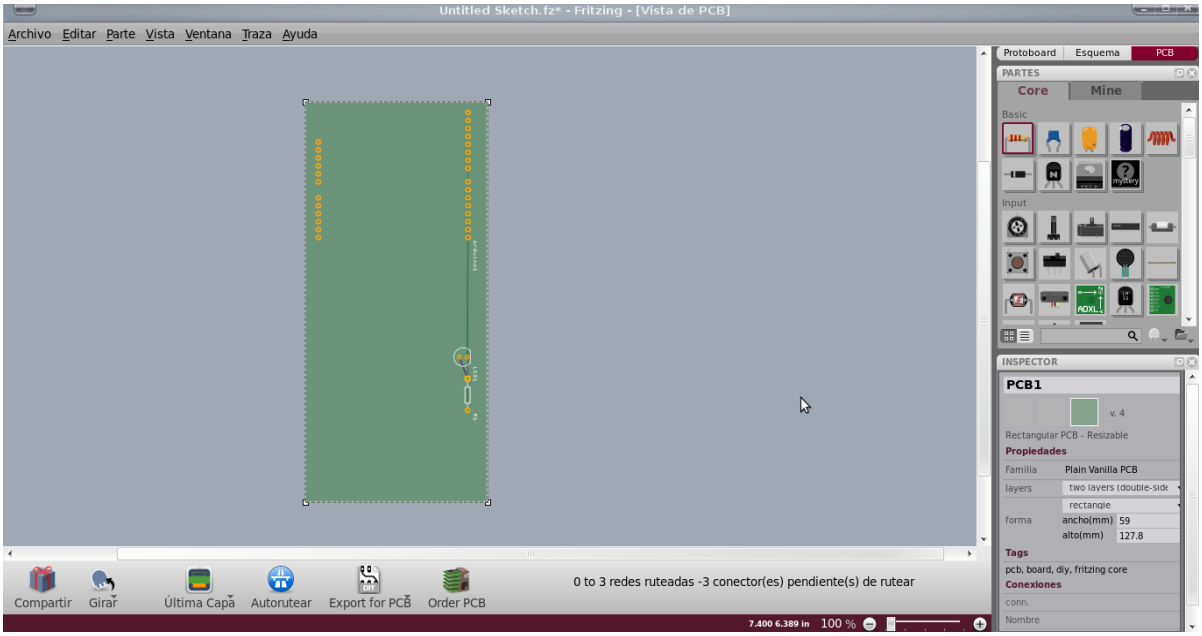
1. Arrastra y suelta un Arduino desde la ventana de Piezas hasta la Vista del Proyecto.
2. Haz lo mismo con la Placa y todas las demás piezas necesarias para armar tu circuito. Si no puedes encontrar una pieza en la librería, usa la Pieza Misteriosa (aquella con el icono de signo de interrogación - ?).
3. Puedes arreglar las piezas seleccionado las, arrastrando las y soltando las, o usando las funciones que se encuentran en la barra de menú, situada debajo de Piezas.
4. Para borrar una pieza, simplemente selecciona y presiona la Barra Espaciadora (BACKSPACE).
5. Selecciona y arrastra el conector Arduino de +5V. Esto debería crear un cable. Coloca el cable en uno de los conectores de la placa. La conexión habrá sido exitosa si observas un circulo o cuadrado verde en los extremos del cable.

Imagen No.3
Creando un circuito



6. Conecta todas las piezas necesarias para que tu circuito luzca como el que construiste en el mundo real. Ten en cuenta que los conectores que no han sido conectados correctamente se verán de color rojo.
7. Si haces un “click sostenido” en cualquier conector, Fritzing sombreara todos los componentes asociados. Esto puede ser especialmente útil cuando quieras ver todas las conexiones asociadas a un componente específico.
8. Puedes curvas los cables solo con añadir puntos de curvatura. Simplemente arrastra en cable en donde quieras curvar lo.
9. Selecciona la pestaña de vista esquemática o PCB para ver o editar el circuito en estas vistas.

Imagen No.4
Vista PCB del circuito

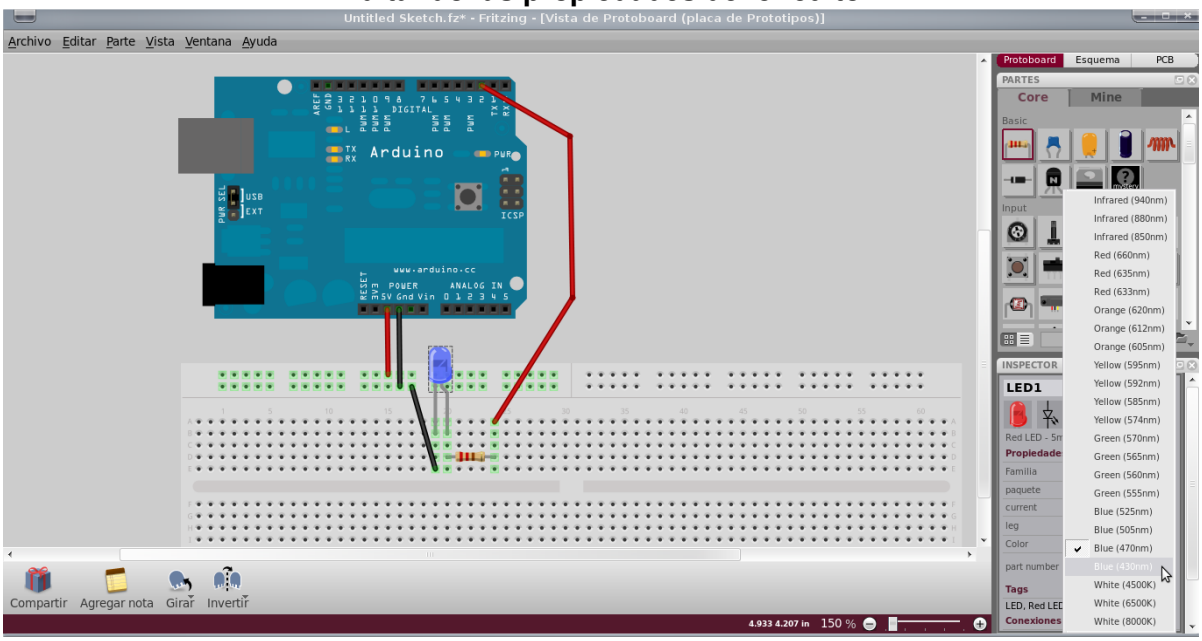


Edición de propiedades

Ahora que tenemos todas las piezas conectadas, veamos como podemos modificar cada una.

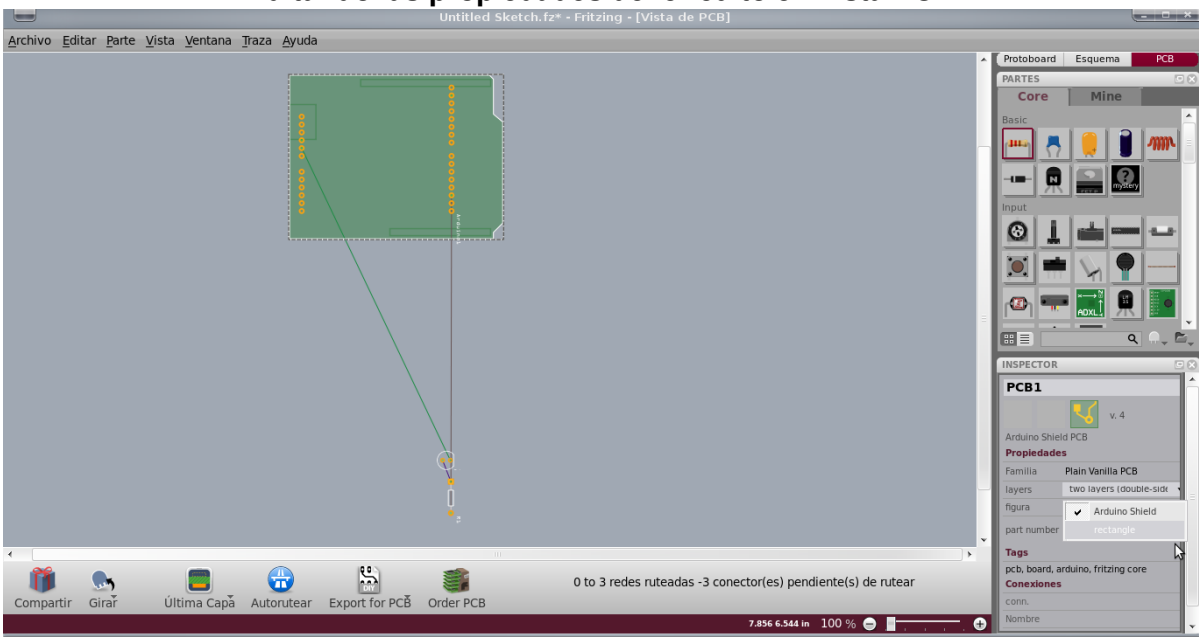
1. Selecciona cualquier pieza del circuito y observa la ventana del Inspector de Piezas.
2. Haz click en el nombre de la pieza y renombrala. Esto es útil cuando quieres distinguir entre piezas iguales.
3. Intenta cambiar otras propiedades.

Imagen No.5
Editando las propiedades del circuito



También puedes modificar las propiedades en la vista PCB. Podrías cambiar la forma de la placa para que luzca como uno de Arduino, un rectángulo de tamaño variable o una forma personalizada.

Imagen No.6
Editando las propiedades del circuito en vista PCB



Exportando un circuito

Luego de terminar de construir el circuito, guarda tu proyecto. Quizás quieras exportar tu circuito a un formato más universal como PDF o una imagen. Para hacer esto, solo debes:

1. Selecciona la vista que deseas exportar (placa, esquemática o PCB).
2. Desde la barra de menú de Fritzing, selecciona Archivo > Exportar > .. y el formato deseado.

Imagen No.7
Exportando el circuito